2001

A Neural Network Approach to Dependent *Reliability Estimation.

Roya Javadpour
Louisiana State University and Agricultural & Mechanical College

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A NEURAL NETWORK APPROACH TO DEPENDENT RELIABILITY ESTIMATION

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Interdepartmental Program in Engineering Science

by
Roya Javadpour
B.S. Industrial Engineering, Isfahan University of Technology, Iran, 1993
M.S. Industrial Engineering, Louisiana State University, LA, 1996
M.S. Engineering Science, Louisiana State University, LA, 2000
August 2001

In the name of God, Most Gracious, Most Merciful

Praise be to God, Lord of the universe. Most Gracious, Most Merciful

Master of the Day of Judgment. You alone we worship. You alone we ask for help.

Guide us in the right path; the path of those whom You blessed;

Not of those who have deserved wrath, nor of the strayers.

"The Key – Al Fatehah"

ACKNOWLEDGMENTS

First and foremost, I would like to thank GOD for all the blessings He has thrown my way. I would also like to take this time to acknowledge those individuals who have been guiding and supporting influences throughout my academic career.

Special thanks are extended to my major professor, Dr. Gerald M. Knapp, for his support, guidance, patience and constructive criticism through the course of my graduate study and preparation of this dissertation. I am also thankful to Dr. Lawrence Mann, Jr., Dr. Michael Kuhl, Dr. Christopher White, and Dr. Robert Lax for being my committee members.

I cannot thank Martha and Joe Juban enough for allowing me to be part of their loving family, helping me through all the rough times and celebrating all my accomplishments with me. They have made my time in Louisiana such an enjoyable memory, which I will always treasure. Along the educational path that I have followed, Martha has offered words of encouragement, doses of reality and most importantly an empathetic ear when I have needed one ..... along with the best chocolate deserts!

My gratitude to Dr. Merrikh Ramazanian and Dr. Mahmood Sabahi, their generosity and kindness will never be forgotten.

The most substantial contributions to my entire education are, of course, from my parents and family who have sacrificed greatly in allowing me to pursue my education far away from home. They have taught me to love learning and let effort and determination lead my way and with their unconditional love and faith,

they have given me the courage to dream. I sincerely dedicate this dissertation to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This research presents the creation of a new model for automating the generation of component and system reliability estimates from simulated field data for tightly coupled systems. The model utilizes the CMAC neural network architecture, which resembles the human cerebellum and is capable of approximating nonlinear functions. An analysis and testing of the network as a tool for reliability prediction of dependent components within an assembly has been performed. In order to evaluate the performance of the model, the network has been tested on simulated data and provided over 90% performance accuracy in learning non-linear functions that represent the dependency between components. This serves as a valuable tool for maintenance personnel faced with important and costly decisions regarding equipment maintenance policies.

CHAPTER 1

INTRODUCTION

For today's sophisticated products and systems, dependability analysis cannot be ignored during the design and operation phases. The trend towards automation and greater complexity has created a need in industry for a comprehensive approach to the development of automated intelligent systems to control, monitor and predict equipment performance.

This research is directed at the creation of a new tool for automating the generation of component and system reliability estimates from field data for tightly coupled systems. In such systems, it cannot be assumed that the component reliability distributions are independent of each other – i.e., the distribution of time to failure (TTF) for one component may be shifted or changed in some way by the age, wear, or other characteristic of other components in the system. In this case, the component reliabilities are said to be dependent. Currently, analysis of dependent reliabilities requires considerable statistical expertise (primarily in correctly making structural assumptions) and professional manpower, both of which are often in short supply. In large military, aviation, and transportation fleets and in large factory and process plants, where reliability is especially critical, the sheer volume of systems and data makes regular analysis / consideration of dependencies impractical.

The remainder of this chapter covers some background information relevant to the work, and further defining and justifying the problem to be dealt with.

1.1 Component Reliability

Reliability can be defined as the ability of an item to perform a required function, under given environmental and operational conditions and for a stated period of time (Hoyland and Rausand 1994). In other words, reliability may be used as a measure of the system's success in providing its function properly. During the life cycle of a component, its time-to-failure cannot be exactly predicted since it represents a random variable characterized by the stochastic properties of the population of potential failure times. Since the item's failure is a stochastic process, the probability of a failure occurring before some specified time t_i is defined by:

$$\int_0^{t_i} f(t)dt \tag{1.1}$$

where $f(t)$ is the component's failure probability density function. This is denoted by $F(t)$ representing the cumulative failure density distribution function, which approaches one when t approaches infinity, indicating that no item can survive failure during an infinite interval of time. The reliability function denoted by $R(t)$ represents the probability that a component will survive at least to a time t, and is defined as:

$$\int_t^{\infty} f(t)dt \tag{1.2}$$

Another reliability measure of a component is its instantaneous failure or hazard rate, denoted by $h(t)$. This hazard rate is an indicator of the rate at which

components will fail at a point in time, given that they have already survived until that time. Mathematically, h*(t)* is determined by (Henly and Kumamoto 1981):

$$h(t) = \frac{f(t)}{(1 - F(t))} = \frac{f(t)}{R(t)} \tag{1.3}$$

.The importance of knowing the failure distribution, reliability and hazard rate functions of a component is due to the fact that design and maintenance decisions are made based upon these factors. Knowing the probability that a failed state may occur by a specific time or in a specific time interval is critical in determining the cost effectiveness of alternative solutions and policies.

The failure distribution of equipment is generally obtained through curve fitting, statistical analysis, and other numerical procedures performed on failure time observation available from historical records. Probability distributions such as the Weibull, negative exponential, normal and lognormal distributions seem to efficiently describe the failure frequency patterns presented by most industrial equipment (Mann 1983, Jardine 1973).

The main objective of a reliability analysis study should be to provide information as a basis for decisions. Reliability technology has a potentially wide range of application areas. Some of these areas are safety/risk analysis, environmental protection, quality, optimization of maintenance and operation, engineering design, and verification of quality/reliability (Hoyland and Rausand 1994).

1.2 System Reliability

Most equipment can fail as a result of the breakdown of any one of its components. The failure data of equipment, if available to product designers or maintenance engineers, is very beneficial. It can help in identifying the different causes of failure of the product and their frequency of occurrence. Consequentially, maintenance engineers can use this information to reduce the failure frequency of the equipment in an efficient manner, thereby leading to a more reliable and efficiently operating system. In addition, this information can be used to design better devices that use some of the same components in different configurations.

Observations of system failure have an important role in decision-making during the design phase of new systems. The reliability of different alternatives needs to be considered before and one of them enters into the development phase where significant investments in production processes will be made. Estimates of equipment reliability need to be made in order to verify that the alternatives can achieve the system reliability requirements. Statistical techniques can be used to determine statistical model or reliability distribution, hazard rate or other characterizations of the system and it's components.

Design or reliability engineers in general perform the reliability prediction of an equipment design by considering the equipment as a system consisting of many components. System reliability prediction is then achieved by (Usher et al, 1990):

1) Compiling a bill of materials,

2) Determining the configuration of the components,

3) Predicting the component reliabilities in the intended operating environment, and.

4) Calculating the system reliability.

In the first step, a list of components that form the system is made. It is quite easy for a design engineer to produce such a list; however, the other steps require more effort.

A piece of equipment or product can be viewed as a system of connected components, any of which can fail. If the components are connected in parallel, then the system will function as long as at least one component is in operational state. However, if components are connected in series, the system will fail as soon as the first of the components fails. In more complicated systems consisting of many components, other configurations, such as x-out-of-k, are possible.

Once a list of components and their configuration information has been collected, component reliability information is needed in order to perform a system reliability prediction. The calculation of predicted system reliability based on this information could be difficult if the failure processes are interdependent. The accurate prediction of component reliabilities under intended operating conditions is arguably the most difficult step in the system reliability prediction process (Usher et al 1990 and Usher et al 1991).

1.3 Equipment Failure

Failure is defined as any incident or condition that causes an industrial system, manufactured product, process, or service to degrade or become unsuitable to

perform its intended function or purpose safely, reliably, and cost-effectively (Witherell 1994).

There can be more than one cause for the failure of any equipment or product. Equipment failure occurs when a component, structure, or system is unable to fulfill its intended purpose, resulting in its retirement from usable service (Alban 1985). Possible failure modes include component deformation, fracture, surface changes (such as cracks), material changes, displacement, leakage, and contaminant (Alban 1985, Lyon 1987). Secondary effects, or symptoms, often occur prior to total machine failure, providing indicators for predicting failure onset.

Failure causes are conditions of the machine and its operation that contributed to a particular failure mode (Alban 1985). Possible failure causes include:

- Design deficiencies, such as inadequate lubrication, or poor bearing design
- Material defects, such as rapid corrosion or wear
- Processing and manufacturing deficiencies
- Assembly errors, such as misalignment or improper tightening of bolts
- Off-design or unintended service conditions
- Maintenance and installation deficiencies
- Improper operation
- Normal wear

Frequently used indicators are vibration signals, noise, heat generation, and particle wear levels as measured in machine lubricants (Knapp and Wang 1992).

1.4 Dependency

There are two major forms of reliability dependency. One type of dependent reliability occurs when the failure of a component or a group of components may change the failure rate of other components in the system. This can be either positive or negative. If the failure of one component leads to an increased tendency for another component to fail, the dependence is said to be positive. On the other hand, if the failure of one component leads to a reduced tendency for another component to fail, the dependence is called negative (Hoyland and Rausand 1994). However, in practical applications positive dependence is usually the most relevant type of dependence. Dependent failures may be classified in three main groups (Hoyland and Rausand 1994):

- *Common failure causes*: Common failure causes are multiple failures that are a direct result of a common or shared root cause.
- *Cascading failures*: Cascading failures are multiple failures initiated by the failure of one component in the system that results in a chain reaction.
- *Negative dependencies*: Negative dependency failures are single failures that reduce the likelihood of failures of other components.

In addition to the above-mentioned groups there are cases where statistical dependencies between component failure distributions exist in a more complex manner. Dependence may be expressed in a components TTF distribution parameters (e.g., shape, variance, location) being a function of characteristics of

other components in the assembly, such as age. Such dependencies may take many forms such as:

- Failure acceleration. Worn components accelerate the wear of other components in the assembly, although component distribution retains essentially the same shape.
- Other changes in distribution parameters.
- Changes to the distribution form itself (for instance, from Weibull to Lognormal).

1.5 Maintenance Planning

Maintenance involves planned and unplanned actions carried out to retain a system in or restore it to an acceptable condition (Sherif and Smith 1981). The purpose of maintenance is to keep the production equipment and plant facilities in the best possible condition in order to produce the best quality products at the lowest possible cost. The primary role of maintenance is to control the condition of the plant equipment and to achieve the following (Wireman 1990):

- Maximum production at the lowest cost, highest quality, and optimum safety levels
- Identify and implement cost reduction
- Provide accurate maintenance and repair records
- Optimize maintenance resources
- Optimize capital equipment life

- Minimize energy usage
- Minimize inventory on-hand

Proper maintenance techniques have been emphasized over the past several decades due to increased complexity of systems, increased quality requirements and rising costs of material and labor. An effective maintenance program should be able to minimize maintenance costs in addition to reducing production loss. Therefore the optimal use of resources requires the decision of when and what resources should be allocated to maintenance. The three main approaches applied in the industry to maintenance are corrective maintenance, preventive maintenance, and predictive maintenance. These approaches are sometimes used in combination in industry (Bloch and Geitner 1983).

Breakdown maintenance is performed only after a failure, which often results in costly damaged equipment and unscheduled downtime in addition to an unsafe environment for the operators. With preventive maintenance, the equipment is inspected periodically by maintenance personnel, regardless of its condition. This process attempts to slow down wear processes leading to failures, and replace components at intervals shorter than their expected useful lifetimes. However, this method does not guarantee that a failure will never occur and the determination of the inspection intervals is not a simple task. It is important for an effective maintenance program to have advance warning of upcoming failures so the necessary planning can be made.

1.5.1 Condition-Based (Predictive) Maintenance

Predictive (or Condition-Based) maintenance involves the intermittent or continuous collection and interpretation of data relating to the operating condition of critical components of the equipment, predicting the occurrence of failure, and determining appropriate maintenance strategies. It is an approach that uses the most cost-effective methodology for the performance of maintenance. The main idea is to ensure maximum operational life and minimum down time within predefined cost, safety, and availability constraints. Predictive maintenance (Bland and Knezevic 1987) is performed as follows:

- Identification of a relevant condition parameter
- Identification of a relevant maintenance parameter
- Determination of the actual conditions under which the components operates
- Performing the tests designed to simulate the actual operating conditions
- Establishing an appropriate maintenance strategy

Predictive maintenance uses direct monitoring of the operating condition, system efficiency, and other indicators to determine the actual mean-time-to-failure or loss of efficiency for each system in the plant and to optimize total plant operation. With a predictive maintenance program the number of breakdowns of all mechanical equipment can be minimized by identifying and detecting the problems before they become serious. Normal mechanical failure modes degrade at a speed directly proportional to their severity, therefore when a problem is detected early major repairs are usually prevented.

A predictive maintenance program schedules specific maintenance tasks as they are actually required by plant equipment, and reduces the number of unexpected failures as well as providing a more reliable scheduling tool for routine preventive maintenance tasks. Therefore, the overall ability of operating systems will be improved. Including predictive maintenance in a management program will provide the ability to optimize the availability of process and greatly reduce the cost of maintenance.

A 1988 survey of 500 plants that have successfully implemented predictive maintenance methods indicates substantial improvements in reliability, availability, and operating costs (Mobley 1990). The survey results show that major improvements can be achieved in maintenance costs, unscheduled machine failures, repair downtime, and spare parts inventory. Mobley identifies the following benefits of predictive maintenance:

- Lower maintenance costs
- Fewer machine failures
- Less repair downtime
- Reduced small parts inventory
- Longer machine life
- Increased production
- Improved operator safety

The goal of a predictive maintenance program is to perform maintenance at the last possible moment prior to failure. The most cost-effective time to maintain a

machine is immediately before it fails or before it starts to lose efficiency. The difficulty is that gathering the amount of data required to accurately time performing predictive maintenance could be a costly process.

There are five nondestructive techniques that are normally used for predictive maintenance management (Mobley 1990):

- Vibration monitoring
- Process parameter monitoring
- Thermography
- Tribology
- Visual inspection

Vibration analysis is the main technique used for predictive maintenance management. With the use of the noise or vibration created by mechanical equipment, this technique determines their actual condition. Another technique is the routine monitoring of process parameters, which should include all machinery and systems in the plant process that can affect its production capacity, and can be accomplished by either manual or microprocessor-based systems. Thermography uses instrumentation designed to monitor the emission of infrared energy, such as heat, to determine the operating condition. By detecting thermal anomalies, the problems within the plant can be located and defined. Lubricating oil analysis, wear particle analysis, and ferrography are tribology techniques used for predictive maintenance. Lubricating oil analysis, as its name implies, is a technique that determines the condition of lubricating oil used in mechanical and electrical

equipment. In wear particle analysis, the particles in a sample of lubricating oil are studied. Whereas lubricating oil analysis determines the actual condition of the oil sample, wear particle analysis provides direct information about the source of this wear. Ferrography is similar to particle wear analysis, the primary difference being that particles are separated by using magnetic field rather than by burning a sample (as in particle wear analysis). Visual inspection of the machinery and systems in a plant is another important technique and may detect potential problems that are missed by the other predictive maintenance techniques.

Predictive maintenance relies on instruments with specific sensors to monitor equipment's normal and abnormal operations and to analyze these signals by comparing them with normal data. Fault diagnosis is the most important step in predictive maintenance. Without accurate detection and identification of machine faults, maintenance cannot be effectively planned and the necessary repair tasks cannot be carried out in time. In other words, equipment monitoring and diagnosis is the first step to a successful predictive maintenance program. Equipment monitoring and diagnosis can be seen as a decision support tool capable of identifying the cause of failure in a machine as well as predicting failure occurrence from a symptom (Wang 1989).

1.5.2 Equipment Fault Diagnosis

Identifying the cause of process abnormalities is important for process automation. Today's world of highly automated, complex systems requires automated maintenance systems to truly accomplish the goals of Computer Aided

Manufacturing (CAM) and Computer Integrated Manufacturing (CIM) and the maintenance function should be involved in the move towards automation and integration of manufacturing systems. The incentives of reduced maintenance costs, increased system availability, and improved productivity and safety have led to an increasing interest in equipment condition monitoring and fault diagnosis. Process monitoring and diagnosis is used in a variety of on line systems in the chemical process, electric power, and other industries in order to quickly detect deviations from acceptable behavior and then accurately diagnose the cause of the deviation so proper corrective actions can be taken. An automated maintenance planning and diagnostics system can provide rapid planning and scheduling response to changing system condition even when faced with enormous quantities of sensory input (Knapp 1992). Fault diagnosis is a vital aspect in the design of operational control systems for large-scale systems with stringent requirements on safety and reliability (Rao and Viswanadham 1987). It is the determination of the primary, independent, fault causes of the process disturbances, through linking of the symptoms generated by them (Becraft and Lee 1993). Incorporation of fault diagnosis capability into the control system enables detection of incipient faults and prevention of their further propagation. This results in an increase in the system availability, reliability and safety, and a reduction in maintenance costs. Fault recognition normally requires detailed analysis of equipment condition data to identify their specific fault pattern. An effective fault diagnostic tool must be able to recognize the characteristics, current conditions, and developing trends of future conditions of operating

equipment, and be able to function under uncertain environments. Diagnosis systems can prevent the unnecessary shutdown of systems when manipulating the operating parameters, can cure irregularities, and can save inspection time by pinpointing the failed component (Chin and Danai 1991).

Equipment fault diagnosis is a popular expert system application area because of the need for an intelligent diagnostic aid, the potential cost savings which could be achieved by the timely alleviation of faults, and the apparent applicability of expert system techniques for diagnosis (Bublin and Kashyap 1989). The timely and reliable detection and diagnosis of failures in technical equipment is important considering operational safety, performance and maintenance economy. Usually, it is desirable to provide warnings and diagnostic information as soon as the failure develops during the functioning of equipment, so that lasting operation under subnormal conditions can be avoided, as well as further deterioration of the equipment and eventual catastrophic breakdowns. This calls for on-line diagnostic techniques, ones that are computationally efficient to the point that they can be run "in parallel" with the physical process (Gertler and Anderson 1992).

1.5.3 Preventive Maintenance

Preventive maintenance refers to those repairs and rebuilds that are scheduled based upon the mean-time-between-failures (MTBF). With a preventive maintenance program, specified components are replaced (usually at regular intervals) when they are becoming worn. It also includes inspection and regular maintenance activities that are planned in order to prevent sudden failure of equipment and to help assure

equipment is operating in satisfactory manner. The main objectives of preventive maintenance include (Niebel 1994):

- To minimize the number of breakdowns on critical equipment.
- To reduce the loss of production that occurs when equipment failures take place.
- To increase the productive life of all capital equipment.
- To acquire meaningful data relative to the history of all capital equipment so that sound decisions as to repair, overhaul, or replacement can be made so as to maximize the return on capital investment.
- To permit better planning and scheduling of required maintenance work.
- To promote better safety and health of the work force.

Equipment inspection is an essential part of a reliability management program. Equipment inspection aims at (Mann 1983):

- Evaluating components in terms of breakdown problems.
- Estimating the occurrence of a breakdown.
- Scheduling repair actions to prevent a major failure.
- Identifying key components that may precipitate a system's failure.

Inspection can be thought of as the middle point between preventive and breakdown maintenance. It applies to both deteriorating systems and hidden failures. The mathematical formulation determines an inspection strategy that will produce the optimal balance between the cost of periodically inspecting the equipment, and the

cost of potential system downtime due to breakdown. Therefore, the objective is to minimize the total cost $C(t_i)$ involved in inspecting the equipment until either a failure is detected or the equipment reaches a failed state. $C(t_i)$ is defined by the following expression:

$$C(t_i) = \frac{\text{Total expected cost per inspection cycle}}{\text{Inspection cycle length}} \quad (1.4)$$

Replacement actions involve the substitution of working equipment before it reaches a failed state at which its functioning is no longer remunerative. Since a stochastic process characterizes the failure time of equipment, the timing of a replacement action is also probabilistic.

When determining the time to perform a replacement, the minimization of the total cost associated with the equipment's operation is usually intended. However, reliability management actions should be given consideration only if the following two conditions are met (Jardine 1973):

- The total cost of the replacement increases after the failure event.
- The equipment presents an increasing failure rate.

If a piece of equipment presents a constant failure rate, replacement before failure will not affect its likelihood of failing again in the next instant. Therefore, its preventive replacement is not useful in economic terms.

In general, the model formulation tries to determine the optimal interval of time between equipment or component replacements so that its total operation and replacement cost per unit time $C(t_r)$ is minimized.

$$C(t_i) = \frac{\text{Total cost in interval } (0,t_i)}{\text{Length of interval}} \qquad (1.5)$$

The complexity of the mathematical model varies according to the assumptions considered in the analysis. Factors such as repairs not returning the equipment to its "good-as-new" state, equipment breakdown during replacement intervals, and inflation considerations can greatly increase the mathematical complexity of the subsequent model. In that case, the model's final result may not be a deterministic interval but rather one with a stochastic nature.

1.6 Problem Statement

Today's complex systems require maintenance personnel to constantly make important and often costly decisions regarding equipment maintenance policies. Usually these decisions are based on practical considerations, previous experiences, historical data and common sense. Reliability analysis, equipment conditions monitoring, and maintenance task scheduling are fundamental parts of a reliability management system. Traditional statistical analytical techniques developed to address these issues require the knowledge of precise information about the form of component functional dependencies, information that is rarely available to industrial practitioners in real life. However, the exact determination of component condition and accurate estimate of system reliability or equipment wear are key factors in maximizing system availability. The monitoring and diagnosis of equipment is a well-established discipline, but much progress remains to be made in the area of automated reliability prediction of systems that can be applied cost-effectively. One

of the main areas of this domain is complex systems that include component interactions. Components in highly coupled assemblies, such as helicopter rotor spindles and truck transmissions are systems that often exhibit these dependencies. Ignoring these factors when calculating assembly system reliability can lead to significant estimation errors. Intensive analysis can, in some cases, be used to determine the form of statistical dependency between coupled components. However, such analysis requires extensive expertise and time.

This work proposes to use Neural Network technology to "learn" the dependency structure of an assembly via automatic learning from historical maintenance data. This knowledge will then be applied to provide a solution to a problem frequently encountered in maintenance of military and commercial aircraft fleets, and in commercial and military vehicle fleets. When an assembly is overhauled with a mix of previously used components, overhauled components, and new components, the assembly's resultant reliability at any future time is a critical issue. The inverse problem will also be solved. That is, considering that multiple units of each component are available (in a mix of used, new, and overhauled states), which units should be selected to create an assembly, so as to maximize assembly reliability (or meet some reliability criteria).

1.7 Objectives

The specific goals of this research may be summarized as follows:

- To develop, validate, and characterize a neural network model for learning the statistical dependency structure of components in coupled assemblies.

- To apply this model, and validate/characterize performance using controlled (simulated) failure data for the prediction of assembly system reliability at any future time t.

CHAPTER 2

LITERATURE REVIEW

In manufacturing the need for improved maintenance has been magnified due to the increased complexity of systems and the shortage of qualified technicians. Downtime costs have risen a great deal because of the increased utilization required to justify the acquisition of expensive equipment. The following sections summarize the related research work of different techniques applied for maintenance and reliability estimation.

2.1 AI Techniques in Maintenance

As industry becomes more automated and sophisticated, it is important that maintenance be seriously considered, carefully planned, and thoroughly implemented. Achieving high productivity in complex systems requires increasing levels of reliability. A broad variety of methods have been developed for diagnosing faults in technical equipment. With the help of these methods, operators may find failures, or causes of abnormalities in equipment, more efficiently and accurately, especially in emergency cases, so that an accident may be controlled. In such a case, human error during decision-making may be reduced significantly. Recent systems have relied upon Artificial Intelligence (AI) techniques to strengthen the robustness of performance. Five AI techniques have been widely applied:

- Expert Systems,
- Model-Based,

- Qualitative Reasoning,
- Fuzzy Logic, and Neural Networks.

2.1.1 Expert and Model Based Systems

There has been an increasing interest in the development of expert systems for fault diagnosis. An expert system is a computer program that functions in a narrow domain dealing with specialized knowledge generally possessed by human experts (Mitra and Pal 1995). When the diagnosis can be presented in precise rules and when all possible malfunctions and their causes are documented, the expert system becomes a tool for troubleshooting machine failures. Common faults are easily captured in this type of system, but it is difficult to guarantee that such purely knowledge based systems are complete and correct. If the knowledge base does not contain the necessary information about a particular fault situation, the expert system will fail and will be unable to diagnosis the fault (Becraft and Lee 1993).

Model-based systems contain a model simulating the structure and function of the machinery. A model-based approach not only requires a description of a system in the static sense, but it also requires a simulation of a dynamic system. They examine a model of the system in order to determine what might have caused the fault; this type of system can discover unusual or unexpected faults but may also suggest unrealistic faults. Model based methods require longer development times and a more detailed knowledge of the process for good performance.

Earlier systems were mostly rule-based expert systems. Expert systems were based on an expert or experienced operator's knowledge, stored as a knowledge base

available to all users. Stone & Webster's Fired Heater Expert System advised on such faults as tube leaks, false drafts, false instrument readings and control valve malfunctions (Keyes 1991), integrating information from a laboratory database, simulation models and statistical models. The TRADES expert system developed for General Motor's Bus and Truck division presents the user with a graphical model of the manufacturing system and guides the user through a series of tests to isolate the faulty component (Keyes 1991). The AMETHYST expert system of Exxon Chemicals is an on-line continuous monitoring system, which monitors the vibration characteristics of the main pumps and compressors of the plant (Keyes 1991).

Model-Based Reasoning and Case-Based Reasoning are popular names for two very different paradigms of artificial intelligence. Both were developed as ways to avoid the proliferation of intertwined "if...then" statements and purely procedural representation of the problem domain in earlier AI based systems. In model-based reasoning, the problem domain can be characterized by an underlying physical model in quantitative or qualitative terms. They are intended to serve much like geometrical axioms, conceptually embodying the entire system. Reasoning from such an underlying model resembles automatic theorem proving more than that of decision tree navigation commonly found in expert systems. Their derivation is guided by the goal of the users and the known facts about the current situation being investigated (Wildeberger 1991). To be effective, model-based reasoning systems require an accurate model of the domain in order to reason about expected behavior.

Models of the domain system must be complete and independent of the reasoning structure of the system in which they are found. Sekine *et al* (1992) discussed the application of model-based fault diagnosis in power systems. Adamovits and Pagurek (1993) used a model, which was developed from object oriented simulation techniques for fault detection and diagnosis of a spacecraft electrical power system. The model consisted of quantitative and qualitative components. The model was constructed by creating objects that defined the individual component characteristics. Instances of these objects were used as building blocks to construct a discrete event, continuous, simulation.

2.1.2 Qualitative Reasoning

Cases where little quantitative information about the process is available, traditional quantitative techniques for analyzing the process cannot be employed and the lack of quantitative data to describe the overall condition of a process becomes a bottleneck. This has lead to the use of qualitative methods, which are concerned with reasoning without precise quantitative information. The importance of qualitative reasoning comes from a need to reason reliably with incomplete knowledge, without having to make unnecessary assumptions or leaving something out (Kuipers 1997). Qualitative and quantitative methods do not contradict each other, thus qualitative reasoning is an attempt to model the knowledge of why and how we do something, it aims at developing models that formalize and implement unclear types of knowledge. It is important to note that qualitative reasoning tries to build a model of both the real existing system and the representation and reasoning

of the person who tries to model this system, therefore a detailed knowledge of the whole process and equipment is required.

Traditionally, models developed were quantitative or mathematical models. Mathematical models of the process and its signals were used for reasoning and fault diagnosis. Models are developed in terms of measurable input and output variables, U(t) and Y(t), respectively (Isermann 1984):

$$Y = f\{U,N,T,X\}$$

Where N represents nonmeasurable disturbance signals, T represents nonmeasurable parameters and X represents partially measurable and non measurable signals. Isermann uses quantitative models (parameter monitoring) for process fault detection in an electrical centrifugal pump and for leak detection in pipelines. Getler and Anderson (1992) studied the detection and diagnosis of failures in physical systems characterized by continuous-time operation. They developed a quantitative diagnosis methodology that utilizes a mathematical model of the physical system. By using several models in parallel and providing for incomplete and/or conflicting inferences, the robustness of the algorithm improved. Zolghadri *et al* (1993) also investigated the problem of failure detection in dynamic systems. Their approach was based on modeling and estimation methods and used some known statistical tools as well as geometric arguments. They applied the idea of using estimated parametric models and confidence regions to detect faults in dynamic systems.

Qualitative reasoning has seen a fast growth in the last decade, from both the simulation and AI communities. The term *qualitative* is normally used with two

different meanings. One refers to the structural properties of the dynamic system: the significant state and input variables, how they relate to each other, where one must close the boundary between phenomena described by the model and those neglected. The second meaning of the word qualitative refers to the magnitude of a variable or the level of influence between different variables. Recent researchers have investigated both these aspects, but the second theme has received greater attention in the artificial intelligence community.

One of the techniques for qualitative analysis is qualitative simulation. In traditional numerical simulation, a set of differential equations over continuous variables is translated into a set of difference equations. Each variable is assigned a real number and the behavior of the system evolves from one time-step to the next time-step according to numerical computations. However, ordinary mathematical equations are not enough for generating casual relations, because polynomial equations do not represent time or causality. In qualitative simulation, the differential equations are translated into a set of constraints over interval valued variables. Each variable is assigned an interval, and because of the ambiguity thus introduced, the behavior of the system branches to include all of the possible evolutions of the system (Walter et al. 1995). The QSIM algorithm developed by Kuipers (1986), provides a framework for developing models in the form of qualitative differential equations (QDEs), which are abstractions of ordinary differential equations, where the values of the variables are described qualitatively and the functional relations between the variables may be known incompletely.

Each variable, including the parameters, is a real-valued function of time with a finite number of critical points. In NSIM (Kuipers 1989, Kay and Kuipers 1993), an extension of QSIM, parameters were assigned lower and upper bounds and functions may be bounded by lower and upper envelopes. QSIM together with NSIM were used to control a nonlinear process in a chemical plant (Gazi *et al* 1994). Another technique for qualitative analysis is Qualitative Process Theory (QPT) which takes the QSIM a step further and reasons about processes and the qualitative equations are associated with particular devices (Forbus 1984).

Vinson and Ungar (1995) utilized Qualitative Modeling and Interpretation (QMI) and Qmimic, a combined monitoring and diagnosis system based on qualitative reasoning, to monitor noisy data streams and diagnose faults from observed dynamic behavior. QMI simulates normal and faulty plant behavior off-line using QSIM models, where as Qmimic incrementally simulates on-line qualitative models which describe the current behavior of the plant. The system was able to diagnose both major faults and small disturbances in a simulated propylene glycol reactor with noisy sensors.

Rao and Viswanadham (1987) used a graph theoretic approach, which is a qualitative technique, for fault diagnosis in dynamical systems. Veerkamp and Hagen (1993) have applied qualitative reasoning for developing an intelligent CAD system. In their system the meta-model represents the design object in terms of function and behavior. A geometric representation is then derived from this meta-model as well as a dynamic representation.

OMISSYS (Opportunistic Multimodel-based Diagnosis System) is an approach to diagnosis where all the available knowledge is framed into different types of models, belonging to space defined by: level of abstraction, approximation/uncertainty, and epistemological type (Bonari and Sassaroli 1993). Knowledge is represented as relations among fuzzy expressions.

Diamon, a model-based troubleshooter based on qualitative reasoning, uses a combined model-based algorithm that integrates monitoring and diagnosis and uses hierarchical models and a dynamic zooming strategy (Lackinger and Wolfgang 1993). Diamon allows many-to-one relations between components and constraints, and thus offers a more expressive qualitative modeling language. Diamon does not use fault models and inductively derived fault hypothesis; it relies on discrepancies with model of correct behavior, which lets it consider unanticipated faults. Trave-Massuyes and Milne (1997) improved condition monitoring for gas-turbine engines by combining qualitative and model-based reasoning. They produce qualitative predictions for values of key variables in the model-based monitoring approach. Their system called TIGER has been operated at the Exxon Chemical Plant and has identified significant faults by integrating several AI technologies. Roddis and Martin (1992) developed a system called CRACK to diagnose fatigue and fracture in steel bridges by combining qualitative reasoning with heuristic knowledge. They verified the system's ability to analyze and predict failures and to critique designs for different kinds of bridges.

Since a single method or approach may not give satisfactory results, researchers are now working in combining techniques to arrive at an efficient model. Hofmann (1993) has developed a diagnostic methodology for off-line diagnosis of a rocket engine, which combines the advantages of model-based techniques based on constraint suspension and of search directed by heuristic expert knowledge. The algorithm enumerates all faults and fault combinations, which are possible, given a fixed set of measurement data in order of their likelihood. Expert knowledge is incorporated into qualitative model-based framework in order to compensate for the lack of sufficient data. The resulting diagnosis is therefore uncertain but represents a best guess consistent with all available data. Oren (1984) developed the concept of multimodels to formalize models containing several submodels, only one of which is put into effect at any given time. Fishwick *et al* (1994) define a multimodel approach as one with a knowledge representation scheme composed of different types of knowledge coupled together. Each type of knowledge provides the basis for a reasoning capability to be used for one aspect of the problem that the overall system addresses. They also discuss about integrating qualitative and numerical simulation models.

2.2 Artificial Neural Networks

Fault diagnosis can be viewed as the process of linking symptoms to causes, paralleling the field of medical diagnosis (Becraft and Lee 1993). Therefore, matching patterns of sensor measurements and process alarms (the symptoms) to specific equipment malfunctions and operational faults (the causes) is the goal of

process fault diagnosis. Neural networks have been successfully utilized in a variety of situations to diagnose system failure (Knapp and Wang 1992, Becraft and Lee 1993, Bernieri *et al* 1994, Lin *et al* 1995). Artificial neural networks are massively parallel interconnections of simple neurons that function as a collective system (Mitra and Pal 1995). Neural networks used for fault diagnosis generally use sensor measurements and process alarms as inputs, and the outputs represent the particular type or category of failure. They have demonstrated good performance in facilitating automatic fault detection and diagnosis in many engineering applications. Neural networks are powerful to use for prediction when precise rules for system diagnosis are not known or when not all possible system failures are known at the time of the construction of the automated fault diagnosis system (Lin *et al* 1995). The primary advantage of neural networks over model-based approaches is that neural networks are trained with available plant data and, theoretically, no prior knowledge of the system is required. In addition, neural networks have proved to be effective in modeling general nonlinear functions (Lee and Kramer 1993, Nelson and Kraft 1995, Carotenuto and Franchina 1996, Kim and Lewis 2000). The main benefit of using artificial neural networks is their ability to diagnose signals that are fuzzy. There has been much effort recently in making a fusion of fuzzy logic and neural networks for better performance in decision-making systems (Carpenter *et al* 1991, Buckley and Hayashi 1993, Marriott and Harrison 1994, Lin *et al* 1995, Mitra and Pal 1995, Uebele *et al* 1995). The utility of fuzzy sets is in their capability in modeling uncertain data that is often encountered in real

life. The uncertainties involved in the input description and output decision are taken care of by the concept of fuzzy sets while the neural net theory helps in generating the required decision region (Mitra and Pal 1995).

Neural networks have now been studied for over 50 years, although they have only become widely used in the last 8-10 years. Artificial neural networks are computer-based systems that attempt to emulate the processing patterns of the human brain (Brown *et al* 1995). In other words, an artificial neural network is a network of many simple processors or units, each possibly having a small amount of local memory. The units are connected by directed communication channels (connections), which carry numeric data. The units operate on their local data and the inputs they receive via the connections and return an output. Due to its new distributed information representation and parallel architecture, artificial neural networks have been applied successfully to all engineering fields such as biological modeling, decision and control, manufacturing and fault diagnosis.

Over the past ten years, much of the progress related to neural network applications has been made in areas of pattern recognition and prediction. Neural networks are a correlational tool, which can be used to adaptively learn associations between fault conditions of complex machinery and fault symptoms exhibited in measurement data, such as vibration signals, oil analysis, or plant performance measures (Knapp *et al* 2000).

Artificial neural networks have several advantages that are desired in manufacturing practice, including learning and adapting abilities, parallel distributed

computation, and robustness. Specific classes of industrial applications for which neural solutions offer advantages with respect to traditional techniques include applications in which algorithmic difficulties arise, considerable computer resources are necessary or theoretical models are lacking. In more detail, the best results using neural network approaches in industrial applications have been obtained when dealing with problems involving (Vena *et al* 1995):

- A large amount of input data.
- A large amount of local computations to be performed in parallel.
- Automatic learning.
- On-line continuous learning.

A number of researchers have studied the application of neural networks to fault diagnosis problems. He *et al* (1992) studied the application of a multilayer feedforward network based machine state identification method. They represented certain fuzzy relationship between the fault symptoms and causes with the high nonlinearity between the input and the output of the network. The procedure for diagnosing the faults is divided into two parts: the design phase, which is performed off-line, and the diagnosis phase, which is performed on-line. They applied the method to a rolling bearing diagnosis problem, which illustrated high correct classification rate and good flexibility. Knapp and Wang (1992) studied the application of a back-propagation network to fault diagnosis of a CNC machine using vibration data. The network was able to classify the fault conditions with 100% accuracy after training on only a small set of examples. Becraft and Lee

(1993) studied the development of an artificially intelligent system as a tool for the diagnosis of faults in large-scale chemical process plants. They integrated two fundamentally different diagnosis techniques, neural networks and expert systems, into a single hybrid architecture system and successfully used it to diagnose faults.

An interesting application of neural fault detection is in dynamic systems. First, the neural network is trained to supply an output for the system under examination in the absence of faults. The detection of an eventual fault situation is made possible by setting the neural network in parallel with the system under control. A fault analyzer detects the signal corresponding to the difference between system and network outputs exceeding a suitable threshold (Bernieri *et al* 1994). Lee and Tan (1993) developed a parallel adaptive neural network control system applicable to nonlinear dynamic systems. Bernieri et al (1994) investigated the possibilities offered by neural networks for system identification and fault diagnosis problems in dynamic systems. Their procedure can be used not only for simple linear systems, but also for complex nonlinear ones and, because of its sensitivity and response time, it is appropriate for on-line fault diagnosis applications.

Neural networks use the process data to learn information about the process problems. Therefore, understanding the content and limitations of the process data used to train the network is important. Farell and Roat (1994) presented neural networks as a part of a fault recognition framework for diagnosing process inefficiencies. Chung *et al* (1994) presented a method of incipient multi-fault diagnosis for large-scale physical systems with complex pipe and instrumentation

such as valves, actuators, sensors, and controllers, by using a modified signed directed graph (SDG) together with the distributed artificial neural networks and a knowledge-based system. Lin et al (1995) utilized neural networks for fault diagnosis of a hydraulic forge. For this purpose, the neural network with 30000-iteration training was used and provided 99% accuracy in identifying causes of the failures of hydraulic forging presses. Knapp *et al* (2000) presented a real-time neural network based condition monitoring system for rotating mechanical equipment. At its core is an ARTMAP neural network, which continually monitors machine vibration data, as it becomes available in an effort to pinpoint new information about machine condition. The system was able to identify the presence of fault conditions with 100% accuracy on both lab and industrial data after minimal training; the accuracy of the fault classification (when trained to recognize multiple faults) was greater than 90%.

Bringing the learning abilities of neural networks to automate and to realize the design of fuzzy logic control systems has recently become an active research area (Carpenter *et al* 1991, Buckley and Hayashi 1993, Lin *et al* 1995, Mitra and Pal 1995, Uebele *et al* 1995, Carpenter *et al* 1995). The integration of neural networks and fuzzy logic systems into a functional system provides a new direction toward the realization of intelligent systems for different applications.

2.2.1 Fuzzy Neural Networks

Fuzzy logic, introduced by Zadeh (1965) in the 1960's is drawing interest as an important AI tool. Due to its ability to classify and cluster patterns based on

approximate values rather than binary logic, it has also been used to implement AI based diagnostic systems. Fuzzy systems operate by testing variables with rules, which produce appropriate responses. Each rule is then weighted by a membership function of the rule invoked, this is a number between 0 and 1, and may be thought of as a probability that a given number is considered to be included in a particular set.

Over the last decade or so, significant advances have been made in two distinct technological areas: fuzzy logic and computational neural networks. The theory of fuzzy logic provides a mathematical framework to capture the uncertainties associated with human cognitive processes. such as thinking and reasoning. On the other hand, the computational neural network paradigms have evolved in the process of understanding the incredible learning and adaptive features of neuronal mechanisms inherent in certain biological species (Gupta and Rao 1994). On a small scale, computational neural networks replicate some of the computational operations observed in biological learning and adaptation. The integration of these two fields, fuzzy logic and neural networks, has created an emerging technological field, the fuzzy neural networks. The fuzzy neural networks have the potential to capture the benefits of the two well-known and often-used fields, fuzzy logic and neural networks.

There is a large and growing body of literature pertaining to the "joining" of neural nets and fuzzy systems. Many of these papers discuss:

- Using a neural net to "tune" the fuzzy system.

- Combining neural nets and fuzzy systems, into so-called hybrid systems.
- Using a neural net to approximate a fuzzy system.

Carpenter *et al* (1992) developed a neural network architecture called fuzzy ARTMAP for incremental supervised learning of recognition categories and multidimensional maps in response to arbitrary sequences of analog or binary input vectors, which may represent fuzzy or crisp set of features. Buckley and Hayashi (1993) show how to construct a hybrid neural net computationally identical to fuzzy expert system. Lin *et al* (1995) addressed the structure and the associated on-line learning algorithms of a feedforward multilayered neural network for realizing the basic elements and functions of a traditional fuzzy logic controller. Mitra and Pal (1995) investigated an application of the fuzzy version of the multilayer perceptron, developed by them, to design a connectionist expert system. Uebele *et al* (1995) discussed a new technique for generating fuzzy rules for pattern classification. Geng and Sheng (1997) studied an automated pattern classification approach based on a neural network structure called the fuzzy cerebellar model arithmetic computer (CMAC) neural network. Javadpour and Knapp (1999) implemented a predictive fuzzy neural network for use as an operator's aid in the diagnosis of faults with high prediction accuracy in an automated manufacturing environment. The overall performance of the network proved to have over 95% accuracy on simulated data and 100% accuracy on real time machine vibration data collected in lab experiments.

2.2.2 Temporal Neural Networks

The applications mentioned so far, mainly focus on the use of neural networks in a steady environment. A specific class of neural network applications is focussed towards dealing with the representation and use of temporal knowledge that base their decision on a set of feature vectors corresponding to different time intervals. Ghosh *et al* (1992) presented the design and evaluation of a comprehensive classification system that uses a hybrid of artificial neural network and statistical pattern recognition techniques for the recognition of short-duration oceanic signals. The problem is complex due to the large variability in both temporal and spectral characteristics even in signals obtained from the same source. Toomarian and Barhen (1992) introduced a new methodology for faster supervised temporal learning in nonlinear neural networks.

Day and Davenport (1993) described an extended form of back-propagation for adapting both weights and time delays in a feed forward neural network, which can be used for signal prediction and spatio-temporal pattern recognition tasks. The delay elements in each of the layers respond to temporal patterns in the hidden node outputs. They show that a network with adaptable delays can achieve smaller errors than a network with fixed delays, having the same number of nodes and connections. Catfolis (1993) described a new implementation of the Real-Time Recurrent Learning (RTRL) algorithm, which is a analogue learning algorithm for fully recurrent networks and uses data described by a temporal stream of inputs and outputs, without time marks which specifies the beginning of a training period. The

implementation increases the performance of the learning algorithm during the training phase by using some a priori knowledge about the temporal necessities of the problem. Levin (1993) introduced a generalization of the layered neural network that can implement a time-varying nonlinear mapping between its observable input and output. She utilizes a neural network architecture called "hidden control neural network" for modeling signals generated by nonlinear dynamic systems with restricted time variability, which allows the mapping to change with time as a function of an additional control input signal. The hidden control input signal provides the ability to capture the non-stationary properties and to learn the underlying temporal structure of the modeled signal (Levin 1993).

Bersini *et al* (1994) investigated Hopfield network transient dynamics for resolving path planning and temporal pattern classification by using learning algorithms for recurrent nets and Lagrangian formalisms. The temporal pattern classification is based on an extension of the Pearlmutter's algorithm, which was developed in order to train temporal trajectories (the time dependent recurrent backpropagation), for the generation of temporal patterns which relies on the use of variational methods. They have applied the algorithm to a simple problem of recognizing five temporal trajectories with satisfactory robustness to distortions. Sotelino *et al* (1994) introduced an algorithm that is able to classify temporal patterns. Their algorithm is also an extension of Pearlmutter's algorithm, and is based on the time trajectory generating the pattern instead of the static pattern itself.

Ghosh and Lubkeman (1995) described an artificial neural network methodology for the classification of waveforms that are captured, as part of a larger scheme to automate the data collection process of recorders. They investigated two different paradigms: the feed-forward neural network (FFNN) and a modification of that, the time-delay neural network (TDNN). A time-delay neural network obtains temporal relationships between features, where features are inputs to the network. It can be viewed as a sparsely connected FFNN with some constraints on the weights, and tends to extract temporal relationships from the input data. It also allows for temporal shifts in the input data without affecting the temporal relationships between the features (Ghosh and Lubkeman 1995). They also presented comparisons of both network paradigms, based on a typical distribution circuit configuration. Dawes *et al* (1995) applied a new diagnostic method to communications network fault diagnosis. Their method used belief propagation to accumulate evidence, which is then used for diagnosis. Successful results were obtained when applied to the accurate, real-time diagnosis of break faults in large wide area data communications networks where the normal status messages provide very uncertain evidence of a fault and its location. Carpenter *et al* (1995) introduced a new neural network architecture for the recognition of pattern classes after supervised and unsupervised learning. The new architecture, called ART-EMAP, extends the capabilities of fuzzy ARTMAP and achieves a synthesis of adaptive resonance theory and spatial and temporal evidence integration for dynamic predictive mapping.

2.2.3 Nonlinear Approximation Neural Networks

In recent years, learning-based control using neural networks has emerged as a practical tool for the control of nonlinear systems with unknown dynamics. They have shown great results in representing the input-output relationships of complex dynamic systems. As a result there have been many demonstrations of the advantages of using neural networks in control systems in which they have been used for approximation of nonlinear systems. Larsen and Cetinkunt (1997) studied a learning controller based on CMAC (cerebellar model arithmetic controller) neural network for servo-control. CMAC is an Associative Memory Neural Network (AMNN) and has been shown to have the capabilities of nonlinear modeling and control. The CMAC proved to be very effective in learning and compensating for the large friction present in the machine tool axes that was highly nonlinear. The results were compared to standard proportional integral derivative (PID) type controllers, which showed that the CMAC algorithm reduced the maximum error by a factor of over 10. Jagannathan (1999) presented a CMAC neural network-based discrete-time controller, which linearizes the unknown multi-input and multi-output nonlinear system through feedback. Control action was defined in order to achieve tracking performance for this unknown nonlinear system.

Lee and Kramer (1993) analyzed machine behavior by developing a pattern discrimination model (PDM) based on a CMAC neural network. A robot was used to study the feasibility of the developed technique. Experimental results showed that the technique was able to analyze machine degradation quantitatively leading to

an innovative methodology to change maintenance practice from breakdown reaction to breakdown prevention. Lin and Wang (1996) implemented a new approach combining CMAC neural network and an advanced trending monitoring method in order to develop an effective fault monitoring and diagnosis tool that was able to estimate fault severity of the system quantitatively.

Nelson and Kraft (1995) explored the real-time control of an industrial robotic arm to balance a mass at the end of a pole. A CMAC neural network was used to learn system nonlinearities and reject any residual noise. Carotenuto et al. (1996) proposed an iterative method to construct a discrete-time nonlinear dynamical system predictor from experimental input-output pairs. Their technique dealt with single-input single-output systems and is suited to work in conjunction with the CMAC neural network. Kim and Lewis (2000) used CMAC neural networks for the application of quadratic optimization for motion control to feedback control of robotic systems. Simulation results from a two-link robot manipulator illustrated the satisfactory performance of the proposed control schemes even in the presence of large modeling uncertainties and external disturbances. Their results showed that the CMAC could cope with nonlinearities through optimization with no preliminary off-line learning phase required. Eldracher et al. (1997) proposed a methodology to improve the function approximation of the CMAC neural network and shorten training time with less memory consumption by using more general, continuous-valued functions. A review of the current neural network literature indicates that the incorporation of artificial neural networks into

the nonlinear function approximation domain yields great benefits in terms of speed, robustness, and knowledge acquisition. We will provide a detailed insight into CMAC's operation in the next chapter.

2.3 System Reliability Estimation

Due to the increased manufacturing competitiveness new methods for reliability estimates are being developed. Reliability plays an important role in many aspects of a system's life cycle. Intelligent manufacturing relies upon accurate component and product reliability estimates for determining optimal maintenance, inspection and replacement schedules. Component reliability distribution estimates can be used in the design and development phase of a system to predict the reliability of a new system design that uses some existing components in a new configuration. This is especially useful when no prototypes are available. These predictions provide critical input in decisions regarding (Usher et al 1991):

- Modification to the design of the system
- Tests to perform on the new design
- Strategies for burn-in of its electronic components
- Reliability performance
- Warranty period offered
- Number and type of spare parts to produce

The earlier such reliability predictions are made in the development stages of the system, the sooner the above-mentioned critical decisions and consequential actions can be made, and the more likely it is that the system will be more reliable.

Usher et al (1991) comment on the timing of system reliability predictions during the design phase that "unfortunately, accurate (system) reliability predictions are extremely difficult to make, especially in the earliest stages of system development, i.e., when few prototypes are available for testing."

Component reliability estimates are also commonly made during the operational phase of a system. Reliability engineers use warranty claims data to make reliability estimates to revise/update warranty claims budget estimates and to obtain an early warning on troublesome parts. This can lead to system design changes, a resulting improvement of the reliability of the system. A production-planning department could use the updated reliability estimates to plan production of the right amount of spare parts and ask the purchasing department to acquire the needed amount and type of raw materials. Alternatively, the purchasing department could use the updated reliability estimates to purchase spare parts manufactured by a third party.

It is clear that the accuracy and timing of component and system reliability estimates are important factors for the economical success of the organization that produces and is responsible for the systems. This can be obtained from the long list of possible uses of component and system reliability distribution estimates during different phases in its life cycle and the financial impact of the decisions based on them. The following are the different sources for component reliability information (Usher et al 1990):

- Reliability test data from the component manufacturer or vendor

- Life tests on the components
- Reliability databases
- Field experience with these components functioning in systems operating in similar environments

These sources are discussed in more detail below.

- Test Data from the Manufacturer or Vendor: The component manufacturer or vendor generally can provide reliability estimates to the purchasing organization. These data are commonly accepted at face value. However, it is often unclear how these estimates have been obtained and how rigorous the components were tested. Furthermore, the exact conditions under which the components were tested likely differ from the conditions under which the components will operate in the intended application.
- Life Tests: Components can be tested under normal operating conditions or subjected to Accelerated Life Tests (ALT). For highly reliable components, such tests can be expensive and/or time-consuming. Elsayed (1996) provides an overview of various models relating the failure data at accelerated conditions to reliability measures at normal stress conditions. The underlying assumption in relating the failure data when using any of these models is that the components operating under normal conditions experience the same failure mechanism as those occurring at the accelerated stress conditions. Though less time-consuming, ALT require costly special facilities to accelerate the life of the component as well as special analytical

skills. For highly reliable components, a large sample size is required to obtain a small number of failure observations used to draw reliability estimates on. Availability or costs of these components might be prohibitive. In addition, individual component life tests might not reflect true component operating conditions (Usher 1996).

- Reliability Databases: The most commonly used sources for reliability prediction are reliability databases (Usher et al 1990). These databases have some drawbacks of their own. The failure rates under various operating conditions that are listed in the military handbooks assume these electronic components exhibit a constant failure rate. This exponential distribution assumption has led to misapplication and critique of the values listed in the handbooks. Failure rates of newly designed systems would be up to one order of magnitude larger than predicted using the handbooks (Evans 1983). Other publications also critique the handbooks (Yellman 1985 and Evans 1988). The accuracy of system reliability predictions based on component reliability estimates from reliability databases is limited. This is caused by the conditions under which the components will be operating as compared to those on which the estimates were based, the assumed type of reliability distribution and the small range of component types considered in the databases.
- Internal Field Experience: Unlike the previously discussed sources, those based on internal field experience generate estimates after assembly of the

components into operational systems. These estimates include the effect the assembly may have on the degradation of the components, as well as any possible effects resulting from the distribution and handling processes that preceded assembly. In addition, human error can contribute to component failure. Real-life data demonstrate the reliability of components in the non-ideal operating environment where failures result from inherent failure processes as well as human errors in distribution, handling, assembly or use of the product. These errors are difficult to model and their effects are difficult to predict, but they need to be considered to accurately predict the reliability of the assembled system.

As mentioned in the previous sections neural networks have been used for numerous applications such as pattern recognition, modeling, signal processing, nonlinear approximations and prediction. However, reliability of industrial systems is an area where the use of neural networks is not yet common. Coit and Smith (1995) formulated and tested a unique approach to the optimization of complex reliability design problems using genetic algorithms and a neural network approximation of the reliability systems. Their approach identified the preferred choice of components and the optimal levels of redundancy for a reliability design problem. The results indicated that using a neural network approximation for system reliability is computationally efficient for optimization problems where calculations of the objective function is impractical. Pasquet and Chatelet (1998) presented a multilayer perceptron neural network to determine the event sequences

that induce the failure of a chemical industrial system and calculate the different parameters of a reliability analysis. Inputs to the network were a numerical value of the input flow into the component and two other were inputs that characterize the state of the component. The output layer of the network represented the output flow of the component. Amjady and Ehsan (1999) proposed a method for the reliability analysis of power systems, which were based on artificial neural networks.

Luxhoj and Shyur (1997) performed a comparison of proportional hazards models and neural networks for reliability estimation. The proportional hazards model is a non-parametric approach developed by Cox (1972), in which a baseline hazard function is modified multiplicatively by covariates. Non-parametric statistical models are used when the failure time data involve complex distributions that are unknown, or when the number of observations is small, which would make the fitting of a failure distribution difficult. However the research conducted by Luxhoj and Shyur proved that a neural network model presents a more accurate technique for using accelerated failure data for estimating reliability at normal operating conditions than does proportional hazards model. Their comparisons were performed for the analysis of time-dependent dielectric breakdown data for a metal-oxide-semiconductor integrated circuit. They used time, temperature and voltage as their inputs and the output was the reliability estimate. Their predictions were then compared with reliability points determined from their Kaplan-Meier estimates, which, however, does not consider the effect of stress loading on the components' life. Another widely used method is multiple regression, in which the applied

stresses are independent variables of the regression model used to predict the time to failure of the individual component. Neural networks are highly nonlinear, and are capable of producing better approximations than multiple regression, which produces a linear approximation (Simpson 1990).

Because of the described drawbacks of the other sources for component reliability estimation in complex systems, this research focuses on the development of a neural network based reliability estimator.

CHAPTER 3

METHODOLOGY

The primary objective of the neural network model is to accurately estimate the dependent reliability of each component in a tightly coupled system. The first step in addressing this problem is to define the neural network architecture and it's prediction environment. In this chapter, the structure and mechanics of the neural network are described in detail and the implementation process is outlined.

3.1 Prediction Environment

Component degradation caused by aging, wear, and environmental effects may decrease reliability of other components and the overall performance reliability of the system. The main problem is to determine the reliability of each component based on the condition of other components with an assembly.

The reliability estimator framework is shown in Figure 3.1. The model framework consists of several neural networks, one for each component. This is to identify the dependent reliability of each component in order to compute the overall system reliability. The accurate prediction of component reliabilities under dependency conditions is the most difficult step in the system reliability prediction process. The following assumptions have be made for this research:

- Since systems with highly coupled assemblies are being considered, a simplifying series-system assumption is made for the description of the configuration. As a result the overall reliability is the product of the individual dependent reliabilities of the components.

- Censored and masked data are not considered.
- Components are dependent on each other.

The design of the neural networks include several stages:

- Description of the component related data
- Formatting the data in training input and target vectors to teach the nonlinear function to the network
- Selection of the network architecture and definition of the learning rules

The goal of this research is to calculate the dependent reliability of each component within an assembly with the minimum information available. This value is dependent on each component's age and base TTF distribution in addition to the age and time to failure of other components in the assembly. Therefore inputs to the network are the age and independent reliability of each component in addition to the time of the desired dependent reliability estimate. The independent reliability is calculated by fitting a distribution to the different times to failure of each component considering the following assumptions:

- Once a component is replaced or overhauled, it is restored to good-as-new condition.
- The times to failure of the components are not dependent on each other.

The output is the dependent reliability of each component. For training purposes the network is provided and trained with a reliability output value that reflects the influence of the different components within the assembly on each other. This is calculated by adjusting the independent time-to-failure of each component through dependency functions explained later in this chapter.

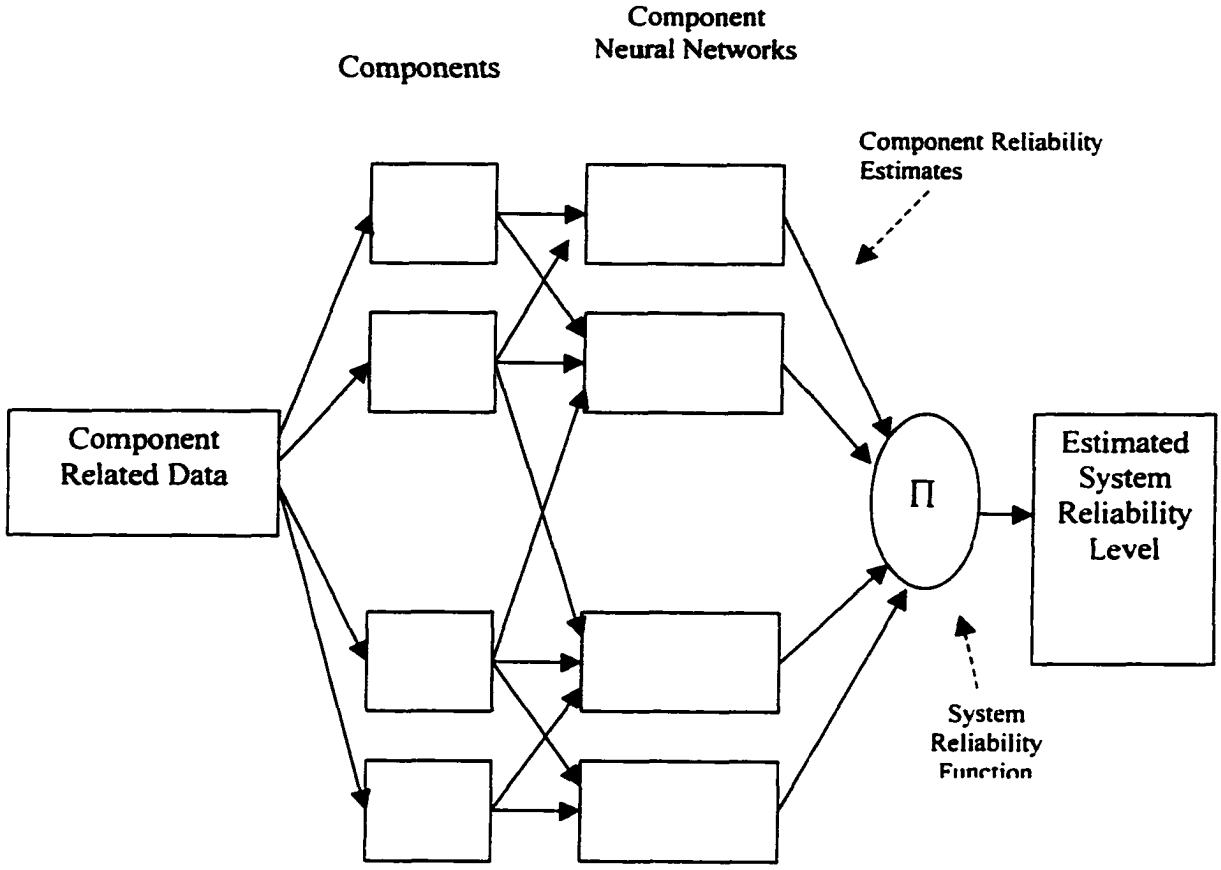

Figure 3.1 Proposed Reliability Estimator Framework

3.2 Network Selection

In selecting a neural network architecture for the application of reliability prediction for dependent components, a network with the ability to represent a variety of complex nonlinear functions over a domain of interest was chosen. In addition to this the following criteria were determined to be critical to success:

- Capability to incrementally adapt over time. As new information becomes available, it is essential that the predicting system be able to quickly adapt to these changes so as to maintain and improve system performance.
- Rapid, stable training. The network should be capable of incorporating new experiences with minimal training time, and without overwriting (unlearning) previous training experiences.

Many common network models have serious disadvantages in these areas. Fast learning makes it possible for a system to adapt quickly to inputs that may occur rarely and that may need immediate and accurate performance. Error-based learning models such as back propagation are not stable in this type of learning environment. Back propagation also requires many iterations to converge and therefore inappropriate for real-time learning. It requires a large number of computations per iteration, which results in a slow algorithm unless implemented on expensive hardware. In general, since all the weights are updated during each learning cycle in feed-forward neural networks, the learning is global and slow. Other network models such as competitive learning cannot learn a temporally stable code in response to arbitrary input environments.

An in-dept review of the current neural network literature generated a promising candidate known as Cerebellar Model Arithmetic Controller (CMAC) initially proposed by Albus (Albus 1975). CMAC is a supervised perceptron-like associative memory network with overlapping receptive fields that has been shown to be capable of approximating nonlinear functions.

Many researchers have focussed their studies on the improvement of the CMAC structure and learning behavior. Lin and Kim (1991) presented a technique that integrated the CMAC into a self-learning control scheme. The modified technique distributively stores learned information, which reduces the required memory and makes the self-learning control scheme applicable to problems of larger size and helps improve the learning speed. They also studied the problem of selecting parameters for the CMAC, which provides a good guideline for parameter selection and was verified by simulations (Lin and Kim 1995). Lin and Chiang (1997) describe the CMAC technique with mathematical formulation and use the formulation to study the CMAC convergence properties. Their investigations proved that the iterative learning in a CMAC structure always converges. Wong and Sideris (1992) also proved that the CMAC learning algorithms always converge with exponential speed under certain conditions. In addition, some of the main properties of CMAC are summarized as the following (Miller et al. 1990):

- CMAC has a built-in local generalization, meaning that input vectors that are "close" in the input space will give outputs that are close, even if the input has not been trained on, as long as there has been training in that region of the state-space.

- CMAC has the property that large networks can be used and trained in practical time. This is because there are a small number of calculations per output even though there are a large number of weights and has a practical hardware realization.

This research focuses on the development of an innovative CMAC prediction system for the reliability of dependent components.

3.3 CMAC Structure and Mechanics

Marr developed a model as showed in Figure 3.2 for the cerebellar cortex, the part of the brain that coordinates fine motor movements (Marr 1969). Based on Marr's model Albus proposed a mathematical model in the mid-1970s as showed in Figure 3.3 and named it Cerebellar Model Articulation Controller (CMAC) (Albus 1975). Analytical processing within the cerebellum suggests that, while there are layers of neurons and large number of interconnects, there is extensive filtering through the use of negative feedback to restrict incoming activity to only a small subset of the most active fibers. The mathematical model resembles the human cerebellum in that inputs excite a fixed number of memory cells and the output is the weighted sum of the response of these cells. The algorithm also preserves the generalization capabilities of the brain in that similar inputs excite many of the same cells resulting in the production of similar outputs (Larsen and Cetinkunt 1997).

The overall operation of the CMAC is briefly described. Figure 3.4 shows a set-oriented overview of the CMAC operations. The input vector is the collection of the measurements required for the prediction of the desired goal. The input space consists of the set of all possible input vectors. The CMAC algorithm maps any

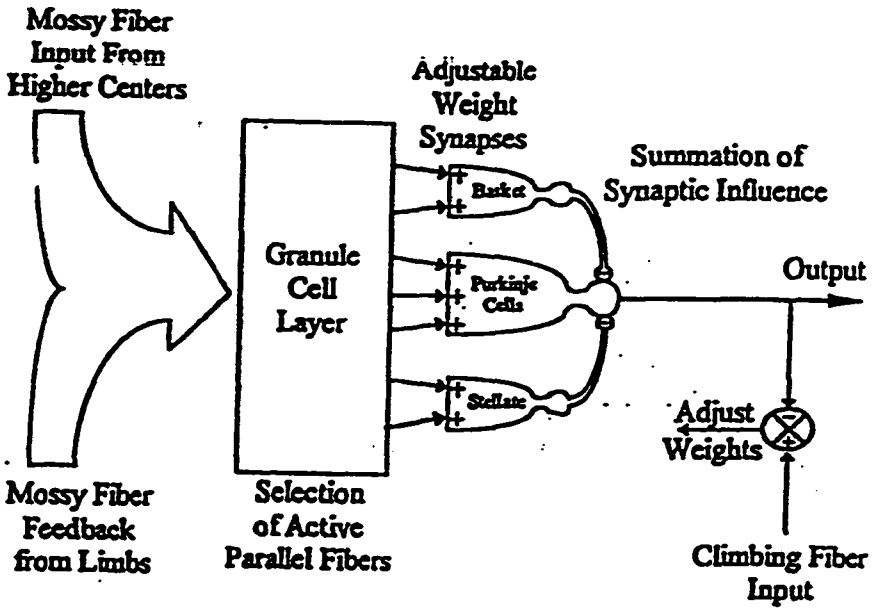

Figure 3.2 A Model of the Cerebullar Cortex (Marr 1969)

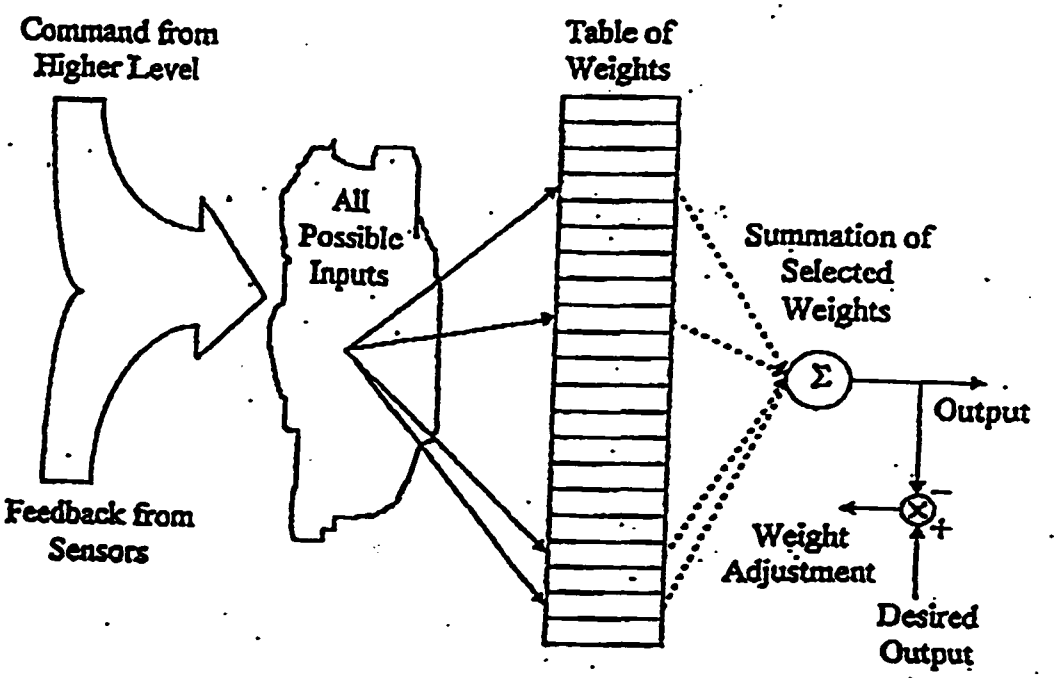

Figure 3.3 The CMAC Model (Albus 1972)

input it receives into a set of C points in a large conceptual memory (A in Figure 3.4). The mapping is done in such a way that two inputs that are "close" in the input space will have their C points overlap in the A memory, with more overlap for closer inputs. If two inputs are far apart in the input space there will be no overlap in their C-element sets in the A-memory. Mapping the A-memory onto a much smaller physical memory A' then reduces the memory requirement. Finally, the contents of the C real memory locations are added to obtain an output. The CMAC is essentially a clever adaptive table lookup technique for representing complex, nonlinear functions over multidimensional, discrete input spaces. It reduces the size of the look-up table and learns the appropriate nonlinear function through a supervised learning process that adjusts the content or weight of appropriate addresses in the look-up table. Since only a small portion of the table is used in obtaining the output and the learning algorithm updates only that portion of the table, the time per learning iteration is significantly reduced. Overall, CMAC consists of a single layer of memory locations, an address-generating unit, and a summing unit and the objective is to approximate a given continuous function so that the distance between the function and its estimate is minimized.

The remainder of this section provides a more detailed insight into CMAC's operation. The overall network architecture is shown in Figure 3.5.

The CMAC network performs a locally weighted approximation of functions by means of some basis functions. The input space is divided into small, overlapped regions called receptive fields (Brown et al. 1993), where the basis functions are defined. For a given input, only the basis functions whose corresponding receptive

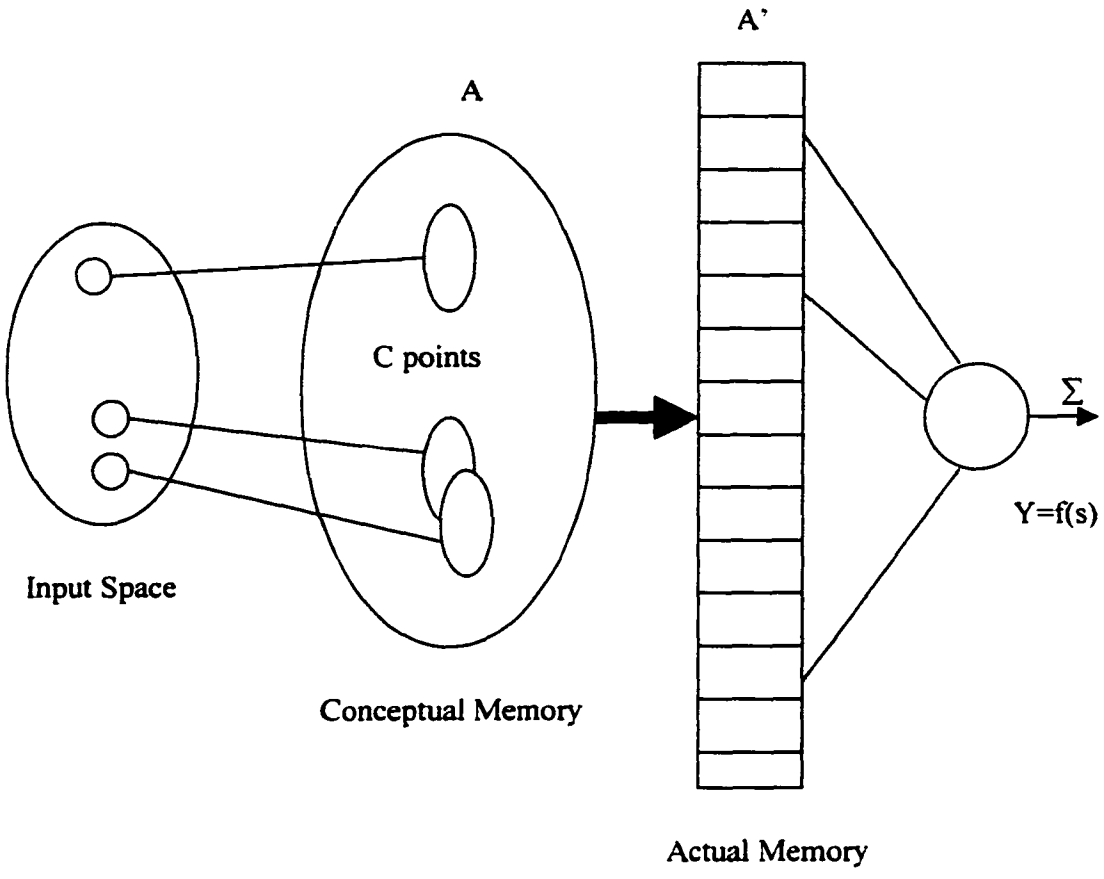

Figure 3.4 Set-Oriented Overview of the CMAC

fields contain that input are activated, and the output is computed as a linear combination of the activated basis function. The original CMAC has constant basis functions, and the receptive fields are hypercubes. Proposed alternatives for basis functions are B-splines, Sigmoid and Gaussian functions (Lane et al 1992, An et al 1993, Chiang and Lin 1996)(Figure 3.6). This is done by a series of mappings:

$$S \rightarrow M \rightarrow A \rightarrow P .$$

where S is the input vector, M is the set of quantization functions (named mossy fibers in the CMAC) used to encode S, A are the elements of some multidimensional memory tables (called granule cells in the CMAC) addressed by M, and P is the resultant output value.

S is a vector composed of N variables each variable may be associated with a measurement parameter. The discrete nature of address generation present in CMAC mapping imposes a form of quantization on the input space, whereby M is partitioned into adjacent disjoint regions, with points in a single region addressing the same set of K weights in CMAC memory (Gonzales-Serrano et al 1998). Therefore any input to the network results in a selection of K active quantization regions, which point at K distinct locations in the network's memory layer (the value of K can be selected by the user). A distinctive feature of the CMAC called "generalization" guarantees that the K memory cells assigned to an input state will have K-1 memory cells overlapping with the memory cells assigned to it's neighboring input state. The generalization reduces the memory requirement in addition to improving the smoothness of function mapping in the sense that similar

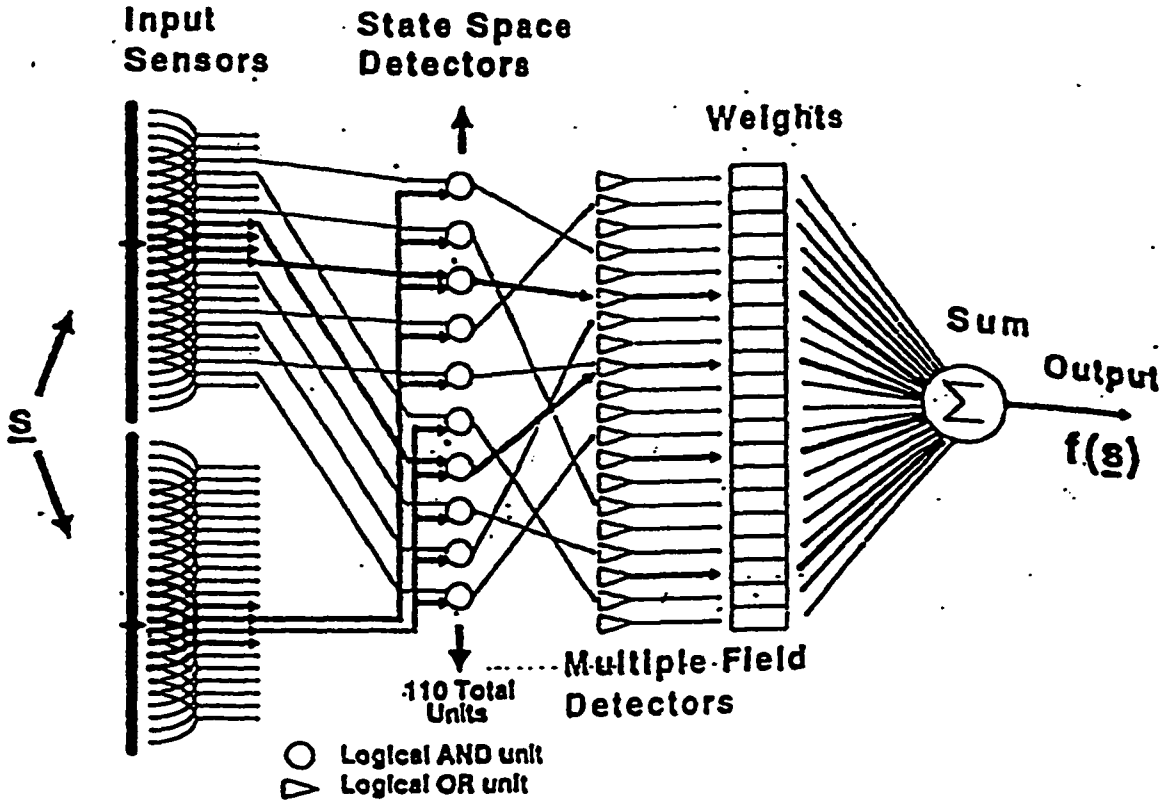

Figure 3.5 Overall CMAC Network Architecture

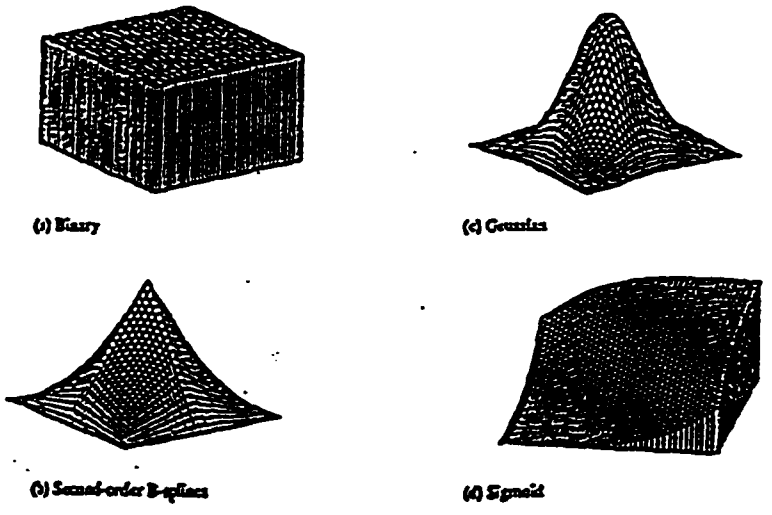

Figure 3.6 Different Basis Functions

input states always generate similar outputs. For each dimension of the input vector, a resolution number R is specified, meaning each input dimension is divided into R equal segments. The value of R varies depending on the application, the higher it is the finer the resolution. The number of partitions per layer is dependent on the number of equal segments in each layer and K that also represents the number of granule cells. The following equation is used to calculate the number of partitions per layer:

P_i = Number of partitions per layer for dimension I

MF = Number of mossy fibers per granule cell

$$P_i = Ceiling\left(\frac{NumSegments_i}{K}\right) \tag{3.1}$$

and

$$MF = \prod_{i=0}^{InputSize-1} Ceiling\left(\frac{NumSegments_i}{K}\right) \tag{3.2}$$

The second mapping, M to A, imitates the structure of granule cells in the cerebellum. It further reduces the physical memory space to a smaller memory space A, where a binary value of the elements in A shows the memory cell activation status (1 for activated and 0 for non-activated):

$$act(s_i) = \begin{cases} 1 & \text{if } s_i \in [0, K-1] \\ 0 & \text{elsewhere} \end{cases} \tag{3.3}$$

The reduced physical memory space is equal to the total number of mossy fibers and calculated by the following equation:

$$K * \prod_{i=0}^{InputSize-1} Ceiling\left(\frac{NumSegments_i}{K}\right) \tag{3.4}$$

Each memory cell location in A is coupled with a weight element in the weight vector W. In other words a granule cell is a weight storing location, or a pointer, pointing to a certain weight value for implementation. Therefore, when two inputs are more than K-1 cells apart, the two weight sets have no members in common. For each input state, there will be K activated elements in W. Finally the real valued output P of the CMAC is calculated by summing the activated weights.

One disadvantage of the original CMAC architecture with basis functions with a binary output (1 if the input lies in the receptive field and 0 otherwise) is that it models a discontinuous function of discrete steps. A smoother surface is more desirable for modeling accuracy. A solution to this is to use other functions such as a normalized Guassian rather than the binary input activating function:

$$\text{act}(s_i) = \begin{cases} \text{gauss}_{ij}(s_i) & \text{if } s_i \in [0, K-1] \\ 0 & \text{elsewhere} \end{cases} \tag{3.5}$$

with

$$\text{gauss}_{ij}(s_i) = \exp\left(-\frac{1}{2}\left(\frac{s_i - \mu_{ij}}{\sigma_{ij}}\right)^2\right) \tag{3.6}$$

where μ_{ij} is the mean and σ_{ij} is the variance. The activation function a_v of a granule cell v is the product of the mossy fiber activation functions, an n-dimensional gaussian. A granule cell becomes active only if all connected mossy fibers are active. When using a non-binary input activating function some regions of the input space are weighted more strongly than others (Figure 3.7). Therefore, the responses of the activated functions require normalization described in Equation 3.6:

$$P = \frac{\sum_j w_j a_j(s)}{\sum_j a_j(s)} \tag{3.7}$$

The CMAC neural network is a supervised learning network. Thus, in the training stage, the desired output of each input vector must be given. In general, the CMAC memory table is initially empty and all the memory elements have a zero value. Such a CMAC would produce a zero response to any input until it has been trained. During training, the CMAC's output for various inputs is compared to the desired output. Therefore, the training consists of adjusting the values in the weight table based on the difference between the present CMAC output and the desired output. For each input-output pair, the weight adaptation process is described in Equation 3.8:

$$w_{jn}=w_{jo} + \beta_1*(\text{ResponseDesired}_j - \text{Response}) + \beta_2(\text{Response} - w_{jo}) \tag{3.8}$$

In the equation above, w_{jn} is the new value of the weight at the address location j, w_{jo} is the old value of the weight at address location j, β_1 and β_2 are learning rates or the percentage of correction, and j = 1,2,.....,K. The weight updating process terminates when the maximum number of learning iterations has been reached. Smaller values of β produce smaller adjustments therefore more training iterations are required. A procedure for learning a function in the CMAC is as follows:

1. Assume F is the function CMAC is to learn. Then $P_d = F(S)$ is the desired input-output relationship.
2. Select n points from the input space where P_d is available, and set the learning iteration L=0.

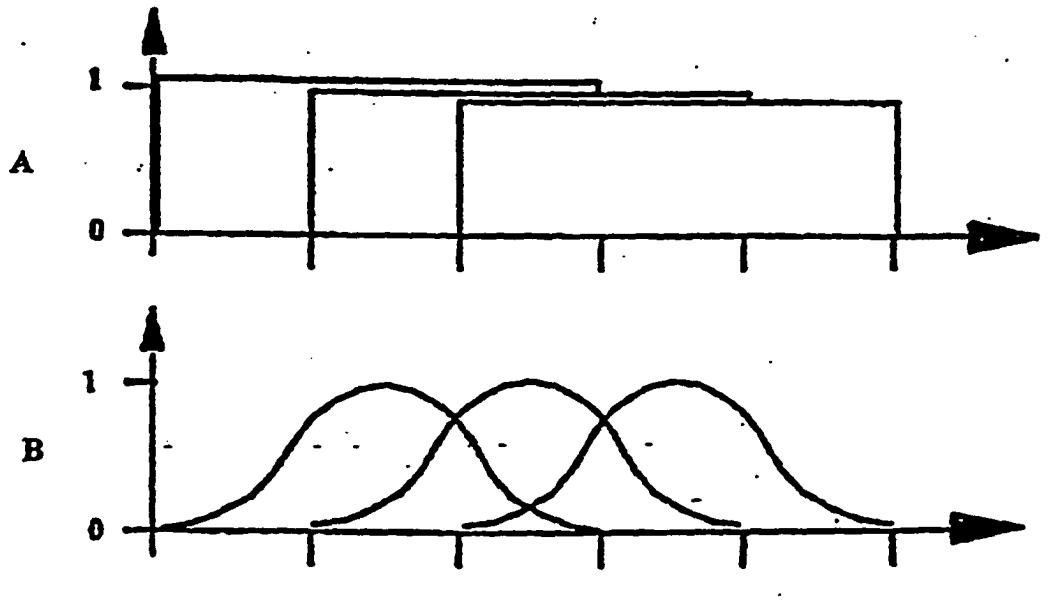

Figure 3.7 Comparison Between The CMAC Binary (A) and Non-Binary (B) Input Activating Functions

3. Calculate the current CMAC output values by entering these n points, P = CMAC(S).
4. For every element in P = (p_1, p_1, ..., p_n) and in P_d = (p_{1d}, p_{2d}, ..., p_{nd}), if L=maximum learning iterations, then stop; the desired values have been stored. Otherwise adjust the weights in the CMAC network using equation 3.7 and set L=L+1.
5. Go to step 3 and continue updating weights until maximum number of iterations has been met.

The data flow of the training method is shown in Figure 3.8. The output of the CMAC neural network for any input is an average over K adjustable weights. During training, the weights are adjusted to reduce the error of the output. However there are a large number of weight values that will produce the same, average and converge to a low error after the training converges

3.4 Network Training

For network training purposes we need an initial estimate of the dependent reliability. For this purpose data sets of different system configurations with similar age and number of overhauls will be grouped together for each component. Grouping is done through the use of the ARTMAP neural network, which is based on Grossberg's Adaptive Resonance Theory (ART) architecture (Grossberg 1988).

ARTMAP is a neural network architecture that performs incremental supervised learning of recognition categories and multidimensional maps in response to input vectors presented in arbitrary order (Carpenter *et al* 1991). It

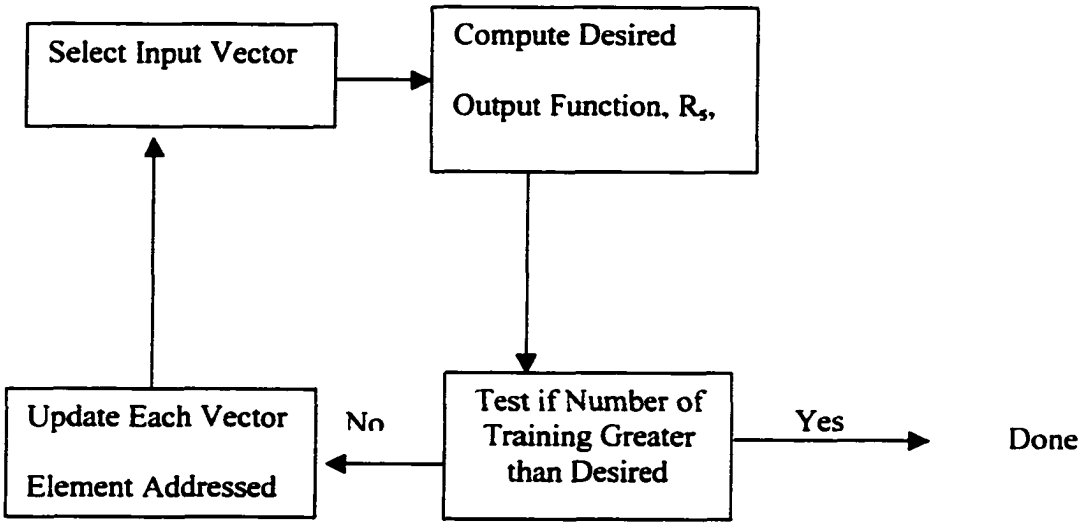

Figure 3.8 Data Flow of the Training Method

classifies inputs by a fuzzy set of features, or a pattern of fuzzy memberships values between 0 and 1, which indicate the extent that each feature is present. The ARTMAP architecture consists of two ART modules, ART_a and ART_b that create stable recognition categories in response to arbitrary sequences of input patterns. The two modules are connected together by a multi-map field, which is a learning network and internal controller that ensures autonomous system operation in real time. The overall architecture is shown in Figure 3.9.

The ART_a Module of the ARTMAP that does unsupervised classification of input data is utilized in this research for grouping purposes. Two main operations are involved in the ART_a module in this process. First, the closest class prototype to the input vector is found through fuzzy set theory operations, which is a measure of match between the input and that category's coded template. Then a vigilance test is performed to control whether if the input will be accepted as a member of the exemplar by testing a measure of each node of the input against each node in the exemplar against a vigilance parameter, which calibrates the minimum confidence that ART_a must have in a hypothesis made by an input, in order to be accepted rather than search for another one.

Input to the network is in the form of a nonnegative vector, $I=(a,a^c)$. The vector is applied to the ART_a input layer F^a_0 and is in the form of complement coding in order to prevent proliferation in the ART network and achieve normalization while retaining information; both the original input a_i and its complement vector $(1-a_i)$ are presented to the network. The elements of the vector

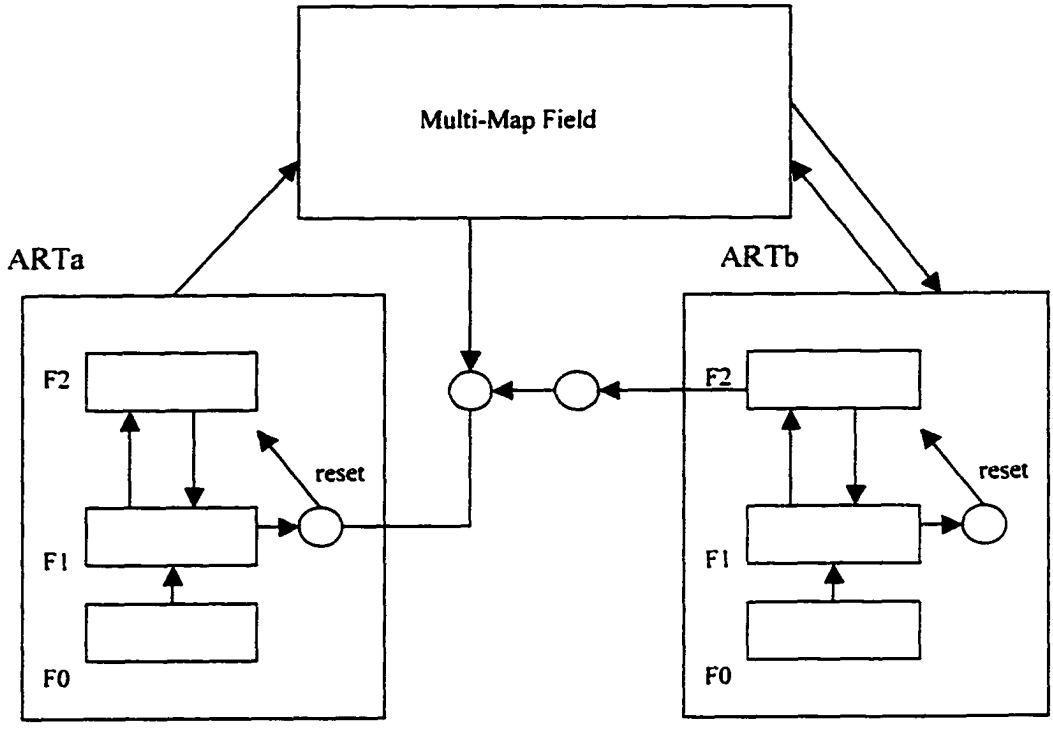

Figure 3.9 Overall Architecture of the ARTMAP

may correspond to any kind of data. For our research the input vectors to the network consist of age and number of overhauls.

Each component of the input vector then corresponds to a single node in the layer F^a_1. The main function of the F^a_1 layer is to perform a comparison between the input value and the hypothesized class exemplar, which is propagated down from the F^a_2 layer. Due to complement coding the input signal to F^a_1 has already been normalized.

The normalized input vector represented by $\mathbf{I}^a$ is then propagated in parallel to layer F^a_2 via weighted bottom-up connections $\mathbf{W}^a_j \equiv (w^a_{j,1}, \ldots, w^a_{j,2M})$ which denote the *j*th ART_a weight vector and are initially equal to one (M is the initial number of elements in the original input vector). The input from F^a_1 to the jth F^a_2 node is give by the category choice function:

$$T_j^a = \frac{|I \wedge w_j^a|}{\gamma + |w_j^a|} \tag{3.9}$$

where the fuzzy AND operator $\wedge$ is defined by :

$$(x \wedge y)_i \equiv \min(x_i, y_i) \tag{3.10}$$

and where the norm | . | is defined by

$$|x| = \sum_{i=1}^{M} |x_i| \tag{3.11}$$

and $\Upsilon > 0$ is a choice parameter and effects the "influence" of the network towards creating new categories or recoding existing ones (Zadeh 1965).

In the grouping process the ARTMAP uses binary choice rule. The node with the largest match score is selected so that only the selected node has positive activation and all the other nodes are zero.

$$y_J^a = \begin{cases} 1 & \text{if} \quad T_J^a > T_j^a \quad \text{for all } j \neq J \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

where y^a_j denotes the F^a_2 output vector. In the case where more than one category choice is maximal, the one with the smallest index is chosen.

In order to control if the input will be accepted as a member of the exemplar, the vigilance test is used. The vigilance test is performed by propagating the signals back down to F^a_1 through the top-down weights. The match function $|I \wedge w_J| / |I|$ of the chosen category J is evaluated against the vigilance criterion:

$$\frac{|I \wedge w_J|}{|I|} > \rho_a \tag{3.13}$$

where $\rho > 0$ is the vigilance parameter. The vigilance parameter defines how well the current input should match a candidate category prototype, and the higher it is, the less variance will be tolerated before a hypothesis is rejected. It also reflects the degree, which I is a fuzzy subset of w_J. If the vigilance test is passed, the initial network hypothesis is judged to be correct and resonance occurs. Resonance is the specific state of the ART network during which learning occurs. Reset occurs if the vigilance test fails; the value of the selected F^a_2 node is set to zero and remains inactive. Then the process is repeated and a new F^a_2 is activated and tested. The search process continues until the vigilance test is passed. When the search ends the weight vector is updated according to the following equation:

$$w^{(new)}_j = \beta(I \wedge w^{(old)}_j) + (1-\beta)w^{(old)}_j \tag{3.14}$$

where $\beta \in [0,1]$ is a learning rate parameter.

The Kaplan-Meier (1958) procedure is utilized to estimate the component's dependent reliability from the grouped simulated data at a certain point in time (observed failure time). The general reliability estimation procedure is as follows:

$$R(t_i) = \begin{cases} 1 & \text{for } i = 0 \\ \prod_{j=1}^{i} \left(1 - \frac{f_j}{n_j}\right) & \text{for } i = 1,2,3,\ldots,k \end{cases} \tag{3.15}$$

where k denotes the number of inspections, f_j the number of components failing in interval (t_{i-1}, t_i), and n_j the number of components that are under the reliability test at the beginning of time t_{i-1}. This research is making the assumption that once a component fails it is replaced, therefore the number of components remain constant throughout the process and the dependent reliability equation is simplified to the following:

$$R(t_i) = \begin{cases} 1 & \text{for } i = 0 \\ \left(1 - \frac{1}{n}\right)^i & \text{for } i = 1,2,3,\ldots,k \end{cases} \tag{3.16}$$

3.5 Operation

Once training is complete, the network is ready to be tested and implemented. The input vector of the network consists of the age and independent reliability of components in addition to the time of the desired dependent reliability estimate. The generated output is the dependent reliability at that time of the considered component. The overall system reliability is then obtained by multiplying the output values of each network together.

3.6 Implementation

The C code for the original CMAC is currently available; we extended and modified these programs for our application. The model is implemented in a Microsoft Access 2000 database (for data storage) using Visual Basics for Applications modules to perform processing tasks. A session run may be initiated from the Session form, which is used to specify run parameters. The following steps are performed in each session run:

1. Generate simulated failure time data.
2. Group Data into closely related categories using ARTMAP.
3. Generate an empirical distribution fit for each group using Kaplan-Meier.
4. Generate an independent distribution fit for each component type using MLE method.
5. Prepare input training and test vectors from the above data and distributions.
6. Train and test the CMAC network.

Parameters governing each of the above steps for a train/test run are stored in the **Sessions** table (it is possible to define multiple session configurations). The following sections describe the above-mentioned steps in more detail.

3.6.1 Generate Simulated Failure Time Data

The GeneratorRawData module is a discrete-event Monte Carlo simulator that simulates the failure performance of a set of identical assemblies operating concurrently. Components in an assembly are assumed to be in series. Upon failure, only the failed part is replaced with a new component. Simulated time-to-failure data is created to obtain the failure distribution and reliability for the network inputs.

The inverse transform technique (commonly applied in simulation) is used to generate random failure times from the Weibull distribution. This method inverts the CDF so that instead of having $U = F(t)$ where U is a probability in the range [0...1] and $F(t)$ is the CDF, one has $t = F(U)$. In this latter form, a random number can be used to derive a random time from the distribution. For the Weibull distribution, the inverse transform is as follows:

$$t = \alpha[-\ln(U)]^{\frac{1}{\beta}} \tag{3.18}$$

where,

t = Random failure rate

α = Characteristic life or scale parameter

β = Slope or shape parameter

U = Random number from zero to one

Simulation data is stored in three tables:

- **SIMCalendar** – calendar of scheduled future events for the simulation engine.
- **SIMComponents** – status information on each component of each assembly, including age.
- **DATA** – records of each failure event of each component of each assembly.

Each failure record in the DATA table consists of the following fields:

- **FailedComponent** – the specific component that failed
- **GID** – Group ID (this field is actually filled in during the next step, "Data Grouping")

- For each component of the failed assembly:
    - **Age** - Component age at time of failure

Several parameters stored in the Session table govern the simulation process. These are:

- **MaxFailures** – the number of failure records to generate before terminating the simulation
- **NumAssemblies** – the number of concurrent assemblies to run in the population.

The components for each assembly and their failure time distributions are specified in the table "Components", and are referentially tied to a session. In pseudocode form the program proceeds as follows:

Initialization

Randomize RN Stream

For i = 1 To NumAssemblies

    Create Assembly i

    Add Components to Assembly i

    Generate initial dependent time of failure for each component

    Find minimum component time-of-failure (i.e., series system)

    Add event to calendar for minimum component time-of-failure

Next i

Simulation Loop

Do While NumFailed < MaxFailures

Get Next Event off the calendar (time, event type, assembly, and component)

Update component states for the assembly

Record the failure to the DATA table

Replace the failed component

Generate next dependent time of failure for each component

Find minimum component time-of-failure

Add event to calendar for min comp time-of-failure

NumFailed = NumFailed + 1

Loop

The time-to-failure (TTF) of each component is dependent not only on it's own base TTF distribution, but also on the age and time-to-failures of other components in the assembly. For each component i in an assembly, this dependency is calculated as:

$$TTFi = (TTFi - age_i) * f \tag{3.19}$$

where TTF_i on the right-side is the TTF generated from the components base distribution, $f(j \neq i)$ is a dependency function, and TTF_i on the left side is the dependent TTF. The following dependency function is utilized to reflect the influence of components on each other:

$$f_I = 1 - \left(\prod_{j \neq i} \frac{age_j}{TTF_j} \right)^x \tag{3.20}$$

where x is a nonnegative normalization factor that effects the influence and degree of dependency of other components in the configuration.

An alternate way to implement the dependency function is to only consider the effect of the component with the shortest time-to-failure. In this case Equation 3.19 is modified to the following:

$$TTFi = (TTF_i - age_j) * 2\left[\left(\frac{1}{1+e^{-[(F-T)/R]}}\right) - 0.5\right] \tag{3.21}$$

where,

$$F = 1 - \max_{i \neq j}\left(\frac{age_i}{TTF_i}\right) \tag{3.22}$$

and

T = Threshold value between -0.2 and 0

R = Rate of drop off (a value between 0.01 and 0.05), the lower it is the steeper the drop off.

Both these dependency functions were applied during model testing.

3.6.2 Data Grouping

The GroupData module creates a text ARTMAP parameter file using parameters defined in the Sessions table, including:

- Coding – whether complement coding is used or not (true/false)
- Alpha – choice function parameter
- Rate – learning rate parameter for ARTMAP
- min_arho – vigilance parameter

Additional ARTMAP parameters written out are either fixed or calculated. These include:

- Max Iterations=1

- Size of input vector = NumComponents
- Size of output vector = 1
- Number of patterns = Number of training patterns = MaxFailures
- Number of testing patterns = 0
- Initial weight values = 1

The module reads the DATA table, and creates an input data file to pass to the ARTMAP program (a command line interface C program). It then spawns the ARTMAP program and passes it the location of the parameter and data files. The ARTMAP program reads these files, processes them to determine categories (groups), and then writes the category (group) data to another text file. The user is asked to press OK to continue the original session run once ARTMAP has completed. The last step in this process includes reading in the ARTMAP output file, and updating the GID field in the DATA table for each failure record with the category determined by the ARTMAP program.

3.6.3 Empirical Fit

The EmpiricalFit module generates a list of groups created by ARTMAP using the SQL query "SELECT DISTINCT FailedComponent, GID FROM Data". For each group it then retrieves all failure data records for the group ordered in ascending time to failure and calculates the Kaplan-Meier estimate (Equation 3.15 and 3.16) for each unique time-to-failure for the group. The final step involves writing the empirical R(t) point (GID, FailedComponent, TTF, R(TTF)) to a table called **EmpiricalR**.

3.6.4 Independent Fit

The IndependentFit module looks up the MaxError parameter for use in the Golden Section Search and fetches a list of component types using an SQL query. For each component type it then performs the following tasks:

- Retrieve all TTF observations for the component
- Find the Maximum Likelihood Estimator's (MLE) for Scale and Shape using a simple Golden Section Search.
- Store the fit scale and shape for the component into the table **IndependentR.**

The MLE method consists of finding the values of alpha and beta that maximize the likelihood function. The likelihood function is the product of the Weibull pdfs for each random failure time in the data set. If it is assumed that there are no censored data, then the likelihood function is (Abernethy 1993):

$$L = \prod_{i=1}^{n} f(t_i) \tag{3.23}$$

where the Weibull pdf is:

$$f(t_i) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-\left(\frac{t_i}{\alpha}\right)^{\beta}} \tag{3.24}$$

The MLE method determines the alpha and beta that maximizes the log likelihood function. This is accomplished by differentiating the logarithm of the likelihood function with respect to alpha and beta. The following function determines the beta parameter for each data set:

$$\frac{\sum_{i=1}^{n} t_i^{\beta} \ln(t_i)}{\sum_{i=1}^{n} t_i^{\beta}} - \frac{1}{n}\sum_{i=1}^{n} \ln(t_i) - \frac{1}{\beta} = 0 \tag{3.25}$$

Then a value is calculated for alpha using the following formula:

$$\alpha = \left(\frac{\sum_{i=1}^{n} t_i^{\beta}}{n} \right)^{\frac{1}{\beta}} \tag{3.26}$$

As can be seen, Equation 3.25 has no closed form solution for β; thus, numeric search methods must be employed. The calculation of the MLE of β requires the iterative calculation of the right-hand portion of the following equation which is derived from equation 3.25:

$$\frac{1}{n}\sum_{i=1}^{n} \ln(t_i) = \frac{\sum_{i=1}^{n} t_i^{\beta} \ln(t_i)}{\sum_{i=1}^{n} t_i^{\beta}} - \frac{1}{\beta} \tag{3.27}$$

This is done through Golden Section Search on the parameter interval between β_{min} = 0.01 and β_{max} = 30. Note that these limits are somewhat arbitrary. They were chosen such that they enclose parameter values in the industrial practice.

As the interval between β_{min} and β_{max} is purposely chosen to be much wider than needed for most shape parameter values observed in practice, and because this will likely lead to an unnecessary reduction in solution speed, an attempt is made to reduce the width of the initial interval. The starting values will be β_{min} = 1.5 and β_{max} = 4.5, and the calculation of the right hand portion value for the limits of the

shape parameter interval are avoided. These calculations are only performed if the starting values indicate that they do not enclose the MLE β.

Iterations should be stopped when estimates are sufficiently close to the MLE values. The stopping criterion used is based on the change in the trial parameter values as compared to the value in the previous iteration. When the difference is less than a small stopping value, ε, then the iterations are stopped as the method is believed to have converged on the MLE for the given data. The stopping value defaults to 10^{-4}. However, the user can specify a different stopping value (10^{-3}, 10^{-5}, or 10^{-6}). Note that the order of the magnitude of the stopping value equals the number of decimal places precision of the resulting parameter estimates (Arts 2000).

In addition this method uses a stopping criterion based on Equation 3.27. If the difference between the left-hand side and the right-hand side values in that equation is less than the stopping value ε for a given trial value of β, then the trial value is considered to be the MLE of β and further trials are halted. This criterion would be met during a Golden Section Search when a trial value in the remaining area of uncertainty accidentally equals the MLE. Therefore, further reduction of the area of uncertainty using Golden Section Search is unnecessary and the search can be halted.

3.6.5 Prepare Vectors

The PrepareVectors module generates the vectors that will be directly input to the CMAC network. It looks up the TimeResolution parameter for use in determining

spacing of generated training and testing vectors, and then create vectors by the following tasks:

For each record in **DATA**:

- Set MaxTime = the maximum time for which an Empirical R(t) estimated is available for this failure records group.
- For k = 0# To MaxTime Step By TimeResolution
  - Create a new vector in the **VECTOR** table.
  - Write out system time to vector
  - Write out age (from **DATA** record) and independent reliability (for system time) of each component to the vector. Independent reliability = exp(-((k/scale)^shape)) where scale and shape are independent reliability parameters for the component from the **IndependentR** table..
  - Using a step function calculate the Empirical R(k) for this vectors group (using the **EmpiricalR** table) and write to the vector as the desired output.
  - If **DATA** record number <=NumTraining, mark this as a training record, else as a testing record
- Next k

Next record

3.6.6 CMAC

The CMAC module implements the CMAC neural network. It reads the following parameters defined in the Sessions table:

- NumCells – number of granular cells
- RFshape – response function shape code
- Beta1 – Learning rate parameter #1
- Beta2 – Learning rate parameter #2
- Iterations – Number of training iterations
- AgeResolution – resolution to be used for age dimensions.
- Resolution – resolution to be used for reliability dimensions
- NumPatterns – total number of patterns
- NumTraining – number of vectors to be used for training

Additional CMAC parameters written out are either fixed or calculated. These include:

- NumTestPatterns = NumPatterns – NumTraining
- Input vector and output vector sizes are calculated based on the number of components

The CMAC module performs the following tasks:

- Initializes the network data structures
- Conducts training:

    For k = 1 To .Iterations

        Retrieves all training records from VECTORS table

        For each record

Set input state and desired response

Get network response

Adjust weights

next record

next k

- Conducts testing:

For k = 1 To .Iterations

Retrieves all training records from VECTORS table

For each record

Set input state and desired response

Get network response

Record desired and actual responses into RESPONSE table.

next record

next k

- Record results to the **Results** table and display a performance report.

CHAPTER 4

PERFORMANCE ANALYSIS

In order to evaluate the performance of the model, the network has been tested using simulated data. The following sections discuss the parameter selection, test efforts, and performance of the model.

4.1 Parameter Selection

There are a number of parameters that can be characterized in the reliability estimator model. There is no deterministic way of selecting a good set of parameters to configure a CMAC network. In most cases the trial-and-error method is used. First the parameters of the ARTMAP network need to be chosen and set. These are the parameters that influence the grouping procedure used to calculate the output that the CMAC is trained with and validated against. One of the ARTMAP parameters addressed in this section is the vigilance parameter. According to the mechanics and structure of the ARTMAP, this parameter changes the sensitivity of the network in the grouping technique. The effect of the vigilance parameter is tested on simulated data of a 3-component assembly. At this stage the network is presented with just an input and the output is the Group-ID to which that assembly belongs. To demonstrate the effect we used a small 3-node network with each node corresponding to the age of the components in the assembly. For comparison of performance the simulated data used consisted of 3 different datasets with the total number of failures being 100, 200, and 300. Our simulations indicate that the vigilance parameter has effective control on grouping. Over the range of 0.1 to 0.5

the network has very low sensitivity, arranging all the inputs into one group. However over the range of 0.6 to 0.9 the number of groups in the network varies from 5 to 25, the higher the vigilance parameter, the more sensitive the network.

The learning rate in the ARTMAP also effects the grouping procedure and needs to be set in order to do unsupervised learning since we are not presenting the network with the desired output to train the vectors against. While testing the different values of the vigilance parameter the network was also tested with different learning rates to see what influence it has on the grouping of components. The training accuracy depends on the level set for the network learning rate as well as the number of learning iterations. The rate of change at each iteration is specified by the learning rate. Higher values of the learning rate result in fast training, which enable the system to adapt quickly to inputs that may occur rarely or have lower iterations values. The network was tested with learning rates between 0.4 and 0.85 in conjunction with the different vigilance parameters. Since the ARTMAP is used for grouping purposes only, the learning iteration is set to 1 requiring a high learning rate. Thus a vigilance parameter equal to 0.85 and learning rate of 0.79 which results in approximately 10 different groups of failure data based on their age was selected and fixed for further experiments.

Two of the important parameters in the CMAC are the resolution and generalization size. The resolution defines the desired segment width. It also specifies the number of segments in each dimension based on the minimum and maximum values of the input vector. The fineness of the resolution parameter sets the size of the smallest change that can occur in the input and still have some effect

on the output and it also determines the breadth of generalization. The generalization size defines the number of layers or granule cells which effect the number of pointers to the weight tables. The combination of these two can determine how fast the CMAC can be trained for a particular function. The larger the number of layers the more training iterations may be required since a smaller portion of the input space is being trained each time. The finer the resolution, the more weights or address pointers required to keep the same range of generalization which may reduce the number of training iterations required. Other parameters that affect the performance of the CMAC are training rates and number of iterations. The purpose of this research is to characterize how changing these parameters influence the accuracy of the network. To demonstrate the effects of these parameters again we considered the same assembly with three components, therefore the size of the input vector would be seven, consisting of age, independent reliability of each component and the time of the desired dependent reliability. It is recommended that the generalization size be equal to the smallest power of 2 greater that (4 * input size). Therefore values of K to be tested are: K= 32, 64, 128, 256

4.2 Performance Analysis on Simulated Data

The performance measure used to validate the network is the prediction accuracy, which is calculated using the following equation:

$$1 - \frac{\sum_{j} \left| (\text{ResponseDesired}_j - \text{Response}_j) \right|}{\sum_{j} \left| \text{Response}_j \right|} \qquad (4.1)$$

where j = 1, 2, 3, ..., K

For the purpose of performance evaluation the following 3 different sizes of datasets were used:

Dataset #1: Number of training failures = 60 Number of testing failures = 40

Dataset #2: Number of training failures = 120 Number of testing failures = 80

Dataset #3: Number of training failures = 180 Number of testing failures= 120

A three-component assembly is considered for the creation of the simulated data. Each data set contains the age and independent reliability of each component as well as the time of the desired dependent reliability estimate. The Weibull distribution was used to create the failure related values in these datasets with the following slope (β) and scale (α) parameters for each component:

$\beta = 3.2$ $\alpha = 1500$

$\beta = 1.5$ $\alpha = 1000$

$\beta = 0.4$ $\alpha = 800$

The slope indicates which class of failures is present (Abernethy 1993):

$\beta < 1.0$ indicates infant mortality

$\beta = 1$ indicates random failures

$\beta > 1$ indicates gradual wearout failures

$\beta > 4.0$ indicates rapid wearout failures

Since each run session creates new simulated data for both training and testing, the results were averaged over 4 runs for each parameter change.

4.2.1 Learning Rate, Training Iterations and Prediction Time

One advantage of the reliability estimator network is that it can be incrementally trained. It learns each input as it is received on-line, rather than performing an off-line learning procedure. Therefore training a new example does not require the presence of the complete set of all known inputs as well as memory and computation resources.

The training time per input is primarily dependent on the number of elements in each vector; however, the training accuracy depends on the level set for the CMAC learning rate parameters. During training, individual weights are adjusted in order to reduce the error in the output. However, there exist a large number of combinations of weight values that result in the same output. After the neural network training converges to a low error, residual error can cause continual small adjustments to the weights. These residual errors may average to zero over time for the CMAC output but not for the individual weights, which may result in some weights having large positive or negative values. These large value weights may cause some problems, which can be fixed by placing a penalty on large weight magnitudes during training as shown in Equation 3.8 in the previous chapter. Literature shows that for good output performance generally β_2 is selected to be at most equal to $\beta_1/4$. For our research purposes the following combination of weights are tested with different training iterations of 50, 80 and 110:

$\beta_1 = 0.2 \qquad \beta_2 = 0.05$

$\beta_1 = 0.4 \qquad \beta_2 = 0.08$

$\beta_1 = 0.6 \qquad \beta_2 = 0.01$

Results of the prediction accuracy from different session runs with different combinations of number of layers (K), learning rates, number of learning iterations (L) and different sizes of training and testing datasets is presented in Tables 4.1 through 4.9. As noted, each prediction accuracy value presented in these tables is an average of results from four run sessions. The generalization capability of the CMAC decreases as the number of layers (K) decrease. The number of layers has to be large enough to allow for accurate storage of the weights. If it is too small too much data is stored over other separately referenced data resulting in the inability to maintain a reasonable level of prediction accuracy. However there is a trade-off between the value of K and the calculation speed, as the K value increases, the reliability estimator needs more calculation time and memory.

Our studies indicate that when K=128 the best test results are achieved, which is consistent over the 3 different sized datasets. When the K value is increased to 256 the prediction accuracy decreases specifying a too broad a generalization. Also increasing the number of training iterations and learning rate does not necessarily improve performance accuracy. Training the network with a low learning rate leads to low prediction accuracy results, increasing the learning rate from 0.4 to 0.6 also shows a decline in performance accuracy. Test results improve from increasing the training iterations from 50 to 80. However increasing the learning iterations to 110 displayed an over-training incident. The following parameters were chosen and set for further experiments of the network model:

$$K = 128 \qquad \beta_1 = 0.4 \qquad \beta_2 = 0.08$$

Table 4.1: Prediction accuracy (%) of dataset #1 and $\beta_1 = 0.2$ $\beta_2 = 0.05$

K	*L=50*	*L=80*	*L=110*
32	62.98	72.89	53.66
64	67.02	84.82	82.41
128	74.37	92.32	85.23
256	56.16	74.67	77.39

Table 4.2: Prediction accuracy (%) of dataset #1 and $\beta_1 = 0.4$ $\beta_2 = 0.08$

K	*L=50*	*L=80*	*L=110*
32	63.57	74.09	65.37
64	82.64	88.42	85.34
128	87.55	93.34	87.20
256	62.16	86.71	82.75

Table 4.3: Prediction accuracy (%) of dataset #1 and $\beta_1 = 0.6$ $\beta_2 = 0.01$

K	*L=50*	*L=80*	*L=110*
32	64.89	69.32	62.77
64	80.72	81.62	80.43
128	82.23	84.94	80.09
256	60.51	79.88	76.12

Table 4.4: Prediction accuracy (%) of dataset #2 and $\beta_1 = 0.2$ $\beta_2 = 0.05$

K	*L=50*	*L=80*	*L=110*
32	61.90	65.18	59.21
64	70.08	80.91	72.56
128	74.13	85.43	82.21
256	69.24	73.21	70.76

Table 4.5: Prediction accuracy (%) of dataset #2 and $\beta_1 = 0.4$ $\beta_2 = 0.08$

K	*L=50*	*L=80*	*L=110*
32	60.51	64.35	65.2
64	72.42	79.46	73.21
128	78.80	89.2	85.74
256	71.43	80.1	79.02

Table 4.6: Prediction accuracy (%) of a dataset #2 and $\beta_1 = 0.6$ $\beta_2 = 0.01$

K	*L=50*	*L=80*	*L=110*
32	59.04	63.12	60.11
64	73.23	74.12	71.76
128	77.08	85.43	82.08
256	70.87	76.20	75.86

Table 4.7: Prediction accuracy (%) of dataset #3 and $\beta_1 = 0.2$ $\beta_2 = 0.05$

K	*L=50*	*L=80*	*L=110*
32	59.32	62.14	60.11
64	76.09	80.24	78.67
128	82.13	84.23	83.90
256	70.05	73.07	71.97

Table 4.8: Prediction accuracy (%) of dataset #3 and $\beta_1 = 0.4$ $\beta_2 = 0.08$

K	*L=50*	*L=80*	*L=110*
32	59.07	62.21	62.04
64	69.43	71.79	68.53
128	79.13	85.84	82.45
256	72.75	75.88	74.89

Table 4.9: Prediction accuracy (%) of dataset #3 and $\beta_1 = 0.6$ $\beta_2 = 0.01$

K	*L=50*	*L=80*	*L=110*
32	60.02	61.23	60.27
64	67.32	69.06	66.87
128	78.11	82.34	80.06
256	69.56	70.21	68.98

As noted before as the values of K, size of the dataset and training iterations increase, the reliability estimator model requires more calculation time. Training and testing time for a dataset #1 with K=128 and L=80 is approximately 90 seconds. This time increases up to 8 minutes for dataset #3 with K=256 and L=80.

4.2.2 Reliability Resolution Size

The results in the previous section were all attained with the reliability resolution factor set to 0.03. The resolution parameter sets the size of the smallest change that can occur in the input and still influence the output. Table 4.10 shows the effect of changing this factor. As predicted smaller resolutions result in better prediction accuracy, however as the resolution gets smaller than 0.03 the change is not significant. In addition test results show that increasing the training iterations does not increase the prediction accuracy, which is consistent with the previous results.

4.2.3 Dependency Functions

The two dependency functions introduced in the previous chapter in Equation 3.19 through Equation 3.22 are used to adjust the time-to-failure of each component based on the age and time-to-failure of the other components within the assembly. Equation 3.20 utilizes a nonnegative normalization factor that affects the degree of dependency of other components in the configuration. Higher values of "x" indicate higher dependency. A value of x = 1 was used for tests discussed in the previous sections. Table 4.11 shows the results of prediction accuracy for different values of "x" for dataset #1, which indicate that the network has the capability to learn the dependencies between the different components within an assembly. As the value

Table 4.10. Prediction accuracy (%) of dataset #1 with different reliability resolutions

Reliability Resolution	*L=80*	*L=110*
0.09	83.88	83.12
0.06	87.67	86.56
0.03	93.34	87.20
0.01	94.01	88.05

of "x" increases the prediction accuracy slightly decreases but still resulting in approximately 90% accuracy.

The second dependency function introduced in Equation 3.21 and Equation 3.22 is based on the component with the shortest time-to-failure rather than all the components. In addition to the shortest time-to-failure this function uses a "threshold value" and "rate of drop off" to calculate the dependencies. The results after changing these values are shown in Table 4.12. Our studies indicate that the network is capable of learning this dependency function with different combination of values for the threshold and rate of drop off, with a reliability accuracy prediction between 82% and 91%.

4.3 Conclusions

Several potential capabilities of the CMAC network have been discussed and shown in this chapter. The properties of CMAC are notable. First, due to its functional interpolation and quantitatively descriptive feature, CMAC may be used as a reliability estimation tool. This overcomes the problem encountered in using other neural network architectures, such as ARTMAP, which perform only pattern recognition. Second, because of the local generalization property of CMAC, input vectors close in input space will look up an overlapping set of weights, therefore similar input patterns will look up almost the same set of weights generating similar outputs. Third, the condensation of large tables into much smaller tables which stores only the limited input values of interest. Finally, the training speed makes on-line component reliability estimation possible.

Table 4.11. Effects of the dependency normalization factor on prediction accuracy (%)

Normalization factor(x)	*L=80*	*L=110*
0.02	92.03	85.8
0.06	92.5	88.16
1	93.34	87.20
2	91.55	86.62
3	89.91	85.23

Table 4.12. Effects of the threshold value and rate of drop off on prediction accuracy (%)

Threshold Value	*R=0.01*	*R=0.03*	*R=0.05*
-0.2	91.3	85.19	81.22
-0.1	89.94	89.60	88.09
0	89.43	84.19	82.12

CHAPTER 5

CONTRIBUTIONS AND FUTURE RESEARCH

5.1 Contributions

This research modeled, implemented and tested a neural network estimator for automating the generation of dependent reliability estimates of components in a tightly coupled system from simulated data. This serves as a valuable tool for maintenance personnel faced with important and costly decisions regarding equipment maintenance policies.

5.2 Future Research

The following issues can be addressed in future research:

- The CMAC architecture can also be tested and compared against the proportional hazards model. This is a mathematical extension of exploratory data analysis, a simple graphical technique for searching for connections between time series data and explanatory factors. In the basic proportional hazards model, a baseline hazard function is modified multiplicatively by covariates as shown in the following:

$$\lambda(t; Z_1, Z_2, \ldots, Z_k) = \lambda_0(t)\exp(\beta_1 Z_1+\beta_2 Z_2+\ldots+\beta_k Z_k) \qquad (5.1)$$

  where, $\lambda(t; Z_1, Z_2, \ldots, Z_k)$ represents the hazard rate at time t, $\lambda_0(t)$ is the baseline hazard rate function, $Z_1, Z_2, \ldots, Z_k$ are explanatory factors (or covariates),and $\beta_1, \beta_2, \ldots, \beta_k$ are the model parameters.

- An optimization process can be developed using the neural network model for selecting a set of component units (out of a set of available units for each component) to construct an assembly so as to maximize assembly reliability.

The optimization technique can be based on a numerical solution. The numerical optimization is in general a search of the input space to choose components to maximize a reliability function and minimize a cost function. In practice, only a discrete number of different combinations can be looked up and evaluated. The time for optimization depends greatly on the desired accuracy. First, the optimization must find the set of components that together will perform with the desired minimum reliability and then, from the set, find the elements that correspond to the highest possible reliability with the lowest cost. Goal programming is a technique that can be used when faced with multiple objectives. In this case, since the relative importance of the goals are not determined precisely, pre-emptive goal programming can be utilized, where the goals can be ranked from most important (maximize reliability) to least important (minimize cost).

REFERENCES

Abernethy, R. B., 1993, The New Weibull Handbook, North Palm Beach, Florida.

Adamovits, P.J., and Pagurek, B., 1993, Simulation (model) Based Fault Detection and Diagnosis of a Spacecraft Electrical Power System, *Proceedings IEEE Ninth Conference on Artificial Intelligence for Applications*, 422-428

Alban, L.E., 1985, *Systematic Analysis of Gear Failures*, (Ohio:American Society of Metals)

Albus, J.S., 1975, A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), *Journal of Dynamic Systems, Measurement, and Control, transactions of ASME*, 220-227

Amjady, N., and Ehsan, M., 1999, Evaluation of Power Systems Reliability by an Artificial Neural Network, *IEEE Transactions on Power Systems*, 14:1, 287-291

Arts, R., 2000, Component Reliability Estimation from Partially Masked and Censored System Life Data Under Competing Risks, PhD Thesis report, Louisiana State University.

Becraft, W.R., and Lee, P.L., 1993, An Integrated Neural Network/Expert System Approach for Fault Diagnosis, *Computers and Chemical Engineering*, 17:10, 1001-1014

Bernieri, A., and D'Apuzzo, M., 1994, A Neural Network Approach for Identification and Fault Diagnosis on Dynamic Systems. *IEEE Transactions on Instrumentation and Measurement*, 43:6, 867-873

Bersini, H., Saerens, M., and Sotelino, L.G., 1994, Hopfield Net Generation, Encoding and Classification of Temporal Trajectories, *IEEE Transactions on Neural Networks*, 5:6, 945-953

Bland, R.J., and Knezevic, J., 1987, A Practical Application of a New Method for Condition Based Maintenance, *Maintenance Management International*, 7, 31-35

Bloch, H.P., and Geitner, F.K., 1983, *Machinery Failure Analysis and Trouble Shooting*, Gulf Pub. Company.

Bonarini, A., and Sassaroli, P., 1993, Opportunistic Multimodel-Based Diagnosis: Framing All the Knowledge We Have to Diagnose Complex Artifacts, *IEEE Transactions on Systems, Man and Cybernetics*, 429-436

Bublin, S., and Kashyap, R.L., 1989, Diagnosing Novel Faults, *2nd International Conf. on Data & Knowledge Systems for Manufacturing Engineering*, Oct, 84-93

Buckley, J.J., and Hayashi, Y., 1993, Hybrid Neural Nets Can Be Fuzzy Controllers and Fuzzy Expert Systems, *Fuzzy Sets and Systems*, 60, 135-142

Carotenuto, R., Franchina, L., and Coli, M., 1996, Nonlinear System Process Prediction Using Neural Network, *IEEE*, 184-189

Carpenter, G.A., Grossberg, S., and Reynold, J., 1991, ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Network. Technical Report CAS/CNS-TR-91-001, Boston University for Adaptive Systems and Department of Cognitive and Neural Systems, Boston, MA, February.

Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., and Rosen, D.B., 1992, Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps, *IEEE Transactions on Neural Networks*, 3:5, 698-713

Carpenter, G.A., and Ross, W.D., 1995, ART-EMAP: A Neural Network Architecture for Object Recognition by Evidence Accumulation, *IEEE Transactions on Neural Networks*, 6:4, 805-818

Catfolis, T., 1993, A Method for Improving the Real-Time Recuurent Learning Algorithm, *Neural Networks*, 6:6, 807-821

Chin, H., and Danai, K., 1991, A Method of Fault Signature Extraction for Improved Diagnosis, *Transactions of the ASME*, 113, 634-638

Chung, H., Bien, Z., Park, J., and Seong, P., 1994, Incipient Multiple Fault Diagnosis in Real-Time with Application to Large-scale Systems, IEEE Transactions on Nuclear Science, 41:4, 1692-1703

Coit, D.W., and Smith, A.E., 1995, Using Neural Network as a Function Evaluator During GA Search for Reliability Optimization, *Proceedings of Intelligent Engineering Systems Through Artificial Neural Networks*, Nov 12-15, 369-374

Cox, D.R., 1972, Regression Models and Life Tables, *Journal of the Royal Statistical Society*, 34, 187-208

Dawes, N., Altoft, J., and Pagurek, B., 1995, Network Diagnosis by Reasoning in Uncertain Nested Evidence Spaces, *IEEE Transactions on Communications*, 43:2-4, 466-476

Day, S.P., and Davenport, M.R., 1993, Continuous-Time Temporal Back-Propagation with Adaptable Time Delays, *IEEE Transactions on Neural Networks*, 4:2, 348-354

Eldracher. M., Staller, A., and Pomple, R., 1997, Adaptive Encoding Strongly Improves Function Approximation with CMAC, *Neural Computations*, 9, 403-417
Elsayed, A.E., 1996, *Reliability Engineering*, (Addison Wesley Longman, Inc.)

Evans, R.A.,1983, Reliability Prediction, *IEEE Transactions on Reliability* R-32, 337

Evans, R.A., 1988, Mil-Hdbk-217D, *IEEE Transactions on Reliability* R-37, 353

Farell, A.E., and Roat, S.D., 1994, Framework for Enhancing Fault Diagnosis Capabilities of Artificial Neural Networks, *Computers and Chemical Engineering*, 18:7, 613-635

Fishwick, P.A., and Zeigler, B.P., 1992, A Multimodel Methodology for Qualitative Model Engineering, *ACM Transactions on Modeling and Computer Simulation*, 2:1, 52-81

Fishwick, P.A., Narayanan, N.H., Sticklen, J., and Bonarini, A., 1994, A Multimodel Approach to Reasoning and Simulation, *IEEE Transactions on Systems, Man and Cybernetics*, 24:10, 1433-1449

Gazi, E., Seider, W.D., and Ungar, L.H., 1994, Control of Nonlinear Processes Using Qualitative Reasoning, *Computers and Chemical Engineering*, 18, S189-S193

Geng, Z.J., and Shen, W., 1997, Fingerprint Classification Using Fuzzy Cerebellar Model Arithmetic Computer Neural Network, *Journal of Electronic Imaging*, 6:3, 311-318

Getler, J.J., and Anderson, K.C., 1992, An Evidential Reasoning Extension to Quantitative Model-Based Failure Diagnosis, *IEEE Transactions on Systems, Man, and Cybernetics*, 22:2, 275-289

Ghosh, A.K., and Lubkeman, D.L., 1995, The Classification of a Power-System Disturbance Wave-Forms Using a Neural Network Approach, *IEEE Transactions on Power Delivery*, 10:1, 109-115

Ghosh, J., Deuser, L.M., and Beck, S.D., 1992, A Neural Network Based Hybrid System for Detection, Characterization, and Classification of Short-Duration Oceanic Signals, *IEEE Journal of Oceanic Engineering*, 17:4, 351-363

Gonzalez-Serrano, F.J., Figueiras-Vidal, A.R., and Artes-Rodriguez, A., 1998, Generalizing CMAC Architecture and Training, *IEEE Transactions on Neural Networks*, 9:6, 1509-1514

Gupta, M.M., and Rao, D.H., 1994, On the Principles of Fuzzy Neural Networks, *Fuzzy Sets and Systems*, 61:1, 1-18

He, Z., Wu, M., and Gong, B., 1992, Neural Network and its Application on Machinery Fault Diagnostics, *IEEE Int. Conf. on Systems Engineering*, 576-579

Henley, E., and Kumamoto, H., 1981, *Reliability and risk Assessment*, (New Jersey: Prentice Hall)

Hofmann, M.O., 1993, Model-Based Diagnosis Directed by Heuristic Search, *Proceedings IEEE Ninth Conference on Artificial Intelligence for Applications*, 197-203

Hoyland, A. and Rausand, M., 1994, *System Reliability Theory, Models and Statistical Methods*, (New York: John Wiley and Sons Inc.)

Isermann, R., 1984, Process Fault Detection Based on Modeling and Estimation Methods - A Survey, *Automatica*, 20:4, 387-404

Jagannathan, S., 1999, Discrete-Time CMAC Neural Network Control of Feedback Linearizable Nonlinear Systems Under a persistence of Excitation, *IEEE Transactions on neural Networks*, 10:1, 128-137

Jardine, A.K.S., 1973, *Maintenance, Replacement, and Reliability* (New York: John Wiley and Sons Inc.)

Javadpour R., and Knapp, G.M., 1999, A Fuzzy Neural Network Approach to Machine Condition Monitoring, *Proceedings of the 25th International Conference on Computers & Industrial Engineering*, March 29-31, 360-365

Kaplan, E.L., and Meier, P., 1958, Nonparametric Estimation from Censored Incomplete Observations, *Journal of American Statistical Association*, 53, 457-481

Kay, H., and Kuipers, B.J., 1993, Numerical Behavior Envelopes for Qualitative Simulation, *Proceedings of the National Conference on Artificial Intelligence (AAAI-93)*, AAAI/MIT Press

Keyes, J., 1991, Manufacturing Survey, *Hand Book of Expert Systems in Manufacturing*, 42-55

Kim, Y.H., and Lewis, F.L., 2000, Optimal Design of CMAC Neural network Controller for Robot Manipulators, *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 30:1, 22-31

Knapp, G.M., 1992, Hierarchical Integrated Maintenance Planning for Automated Manufacturing Systems, *PhD. Thesis Report*, University of Iowa, Iowa.

Knapp, G.M., Wang, H.P., 1992, Machine Fault Classification: A Neural Network Approach, *International Journal of Production Research*, 30:4, 811-823

Knapp, G.M., Javadpour, R., and Wang, H.P., 2000, An ARTMAP Neural Network Based Machine Condition Monitoring System, *Accepted to be published in Journal of Quality in Maintenance (forth-coming)*

Kuipers, B.J., 1986, Qualitative Simulation, *Artificial Intelligence*, 29, 289-338

Kuipers, B.J., 1989, Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge, *Automatica*, 25, 571-585

Kuipers, B.J., 1997, Using Qualitative Reasoning, *IEEE Expert*, 94-97

Lin, C.S., and Chiang, C.T., 1997, Learning Convergence of CMAC Technique, *IEEE Transactions on Neural Networks*, 8:6, 1281-1292

Lin, C.S., and Kim, H., 1991, CMAC-Based Adaptive Critic Self-Learning Control, *IEEE Transactions on Neural Networks*, 2:5, 530-533

Lin, C.S., and Kim, H., 1995, Selection of Learning Parameters for CMAC-Based Adaptive Critic Learning, *IEEE Transactions on Neural Networks*, 6:3, 642-647

Lin, C.C., and Wang, H.P., 1996, Performance Analysis of Rotating Machinery Using Enhanced Controller (E-CMAC) Neural Networks, *Computers and Industrial Engineering*, 30:2, 227-242

Lackinger, F., and Wolfgang, N., 1993, Diamon: A Model-Based Troubleshooter Based on Qualitative Reasoning, *IEEE Experts*, 33-40

Larsen, G., and Cetinkunt, S., 1997, Low Speed Motion Control Experiments on a Single Point Diamond Turning Machine Using CMAC learning Control Algorithm, *Journal of Dynamic Systems, Measurement, and Control*, 119, 775-781

Lee, J., and Kramer, B.M., 1993, Analysis of Machine Degradation Using a Neural Network Based Pattern Discrimination Model, *Journal of Manufacturing Systems*, 12:5, 379-387

Lee, T.H., and Tan, W.K., 1993, Real-Time Parallel Adaptive Neural Network Control For Nonlinear Servomechanisms - An Approach Using Direct Adaptive Techniques, *Mechauonics*, 3:6, 705-725

Levin, E., 1993, Hidden Control Neural Architecture Modeling of Nonlinear Time-Varying Systems and its Applications, *IEEE Transactions on Neural Networks*, 4:1, 109-116

Lin, H., Yih, Y., and Salvendy, G., 1995, Neural Network Based Fault Diagnosis of Hydraulic Forging Presses in China, *International Journal of Production Research*, 33:7, 1939-1951

Lyon, R.H., 1987, *Machinery Noise and Diagnostics*, (Boston, MA: Butterworth)

Luxhoj, J.T. and Shyur, H., 1997, Comparison of Proportional Hazards Models and Neural Networks for Reliability Estimation, *Journal of Intelligent Manufacturing*, 8, 227-234

Mann, L., 1983, *Maintenance Management*, (Lexington, MA: Lexington Books)

Marr, D., 1969, A Theory of Cerebellar Cortex, *Journal of Physiol.*, 202, 437-470

Marriott, S., and Harrison, R., 1995, A Modified Fuzzy ARTMAP Architecture for the Approximation of Noisy Mappings, *Neural Networks*, 8:4, 619-641

Miller, W.T., Glanz, F.H., and Kraft, L.G., 1990, CMAC: An Associative Neural Network Alternative to Back-propagation, *Proceedings of the IEEE*, 78:10, 1561-1567

Mitra, S., and Pal, S.K., 1995, Fuzzy Multi-Layer Perceptron, Inferencing and Rule Generation, *IEEE Transactions on Neural Networks*, 6:1, 51-63

Mobley, R.K., 1990, *An Introduction to Predictive Maintenance*, (New York: Van Nostrand Reinhold)

Moore, T.N., Jeswiet, J., and Lipsett, M.G., 1992, Machinery Fault Diagnosis for Automated Machining Systems, *The Journal of Applied Manufacturing Systems*, Fall 1992, 35-42

Nelson, J., and Kraft, G.L., 1995, Using CMAC Neural Networks and Optimal Control, *IEEE*, 2386-2390

Niebel, B.W., 1994, *Engineering Maintenance Management*, (Marcel Dekker, Inc.)

Oren, T.I., 1984, Model-Based Activities: A Paradigm Shift, *Simulation and Model-Based Methodologies: An Integrative View*, T.I. Oren, B.P. Zeigler, and E.M.S., Eds. (Springer-Verlag, New York), 3-40

Pasquet, S. and Chatelet, E., 1998, How to Use Neural Networks to Study the Reliability of Dynamic Systems, *IEEE*, 226-230

Rao, S.V.N., Viswanadham, N., 1987, Fault Diagnosis in Dynamic Systems: A Graph Theoretic Approach, *Int. Journal of Systems Science*, 18:4, 687-695

Roddis, W.M.K., and Martin, J.L., 1992, CRACK – Qualitative Reasoning About Fatigue and Fracture in Steel Bridges, *IEEE*, 41-48

Sherif, Y.S., and Smith, M.L., 1981, Optimal Maintenance Models for Systems Subject to Failure - A Review, *Naval Research Logistics Quarterly*, 28:1, 47-65

Sekine, Y., Akimoto, Y., Kunugi, M., Fukui, C., and Fukui, S., 1992, Fault Diagnosis of Power Systems, *Proceedings of IEEE*, 80:5, 673-683

Simpson, P., 1990, *Artificial Neural Systems*, (Pergamon Press, New York)

Sotelino, L.G., Saerens, M., and Bersini, H., 1994, Classification of Temporal Trajectories by Continuous-Time Recurrent Nets, *Neural Networks*, 7:5, 767-776

Toomarian, N.B., and Barhen, J., 1992, Learning a Trajectory Using Adjoint Functions and Teacher Forcing, *Neural Networks*, 5:3, 473-484

Trave-Massuyes, L., and Milne, R., 1997, Gas-Turbine Condition Monitoring Using Qualitative Model-Based Diagnosis, *IEEE*, 22-31

Uebele, V., Abe, S., and Lan, M.S., 1995, A Neural Network-Based Fuzzy Classifier, *IEEE Transactions on Systems, Man, and Cybernetics*, 25:2, 353-361

Usher, J.S., Alexander, S.M., and Thompson, J.D., 1990, System Reliability Prediction Based on Historical Data, *Quality and Reliability Engineering International* 6, 209-218

Usher, J.S., Alexander, S.M., and Thompson, J.D., 1991, Predicting the Reliability of New Products at IBM, *Proceedings of the 1991 Annual Reliability and Maintainability Symposium*, 208-213.

Usher, J.S., 1996, Weibull Component Reliability-Prediction in the Presence of Masked Data, *IEEE Transactions on Reliability*, 45, 229-232.

Veerkamp, P., and Hagenm P.T., 1993, Qualitative Reasoning About Design Objects, *Robotics & Computer-Integrated Manufacturing*, 10:1/2, 33-39

Vena, L.D., Mastretta, M., and Ricciardiello, L., 1995, Neural Network Architectures for Industrial Applications, *Biosensors & Bioelectronics*, 10, 231-236

Vinson, J.M., and Ungar L.H., 1995, Dynamic Process Monitoring and Fault Diagnosis with Qualitative Models, *IEEE Transactions on Systems, Man, and Cybernetics*, 25:1, 181-189

Wang, S.S., 1989, AI and Expert Systems for Diagnosis, *Proceedings of the 1st International Machinery Monitoring and Diagnosis Conference and Exhibit*, Las Vegas, NV, September, 516-522

Wilberger, A.M., 1991, AI & Simulation, *Simulation*, 57:3, 148

Winston, W.L., 1991, *Introduction to Mathematical Programming Applications and Algorithms*, (Pws-Kent Publishing Company, Boston)

Witherell, C.E., 1994, *Mechanical Failure Avoidance, Strategies and Techniques*, (McGraw-Hill Inc., New York, NY)

Wong, Y., and Sideris, A., 1992, Learning Convergence in the Cerebellar Model
Yellman, T.W., 1985, Comment on Reliability Prediction, *IEEE Transactions on Reliability* R-34, 504-506.

Zadeh, L., 1965, Fuzzy Sets, *Information and Control*, 8, 338-353

Zolghadri, A., Bergeon, B., and Monsion, M., 1993, A Two-Ellipsoid Overlap Test for On-Line Failure Detection, *Automatica*, 29:6, 1517-1522

VITA

Roya Javadpour was born in Shiraz, Iran, in 1970, the second daughter of Mahin Farahi and Seyyed Hossein Javadpour. She received her bachelor of science in industrial engineering from Isfahan University of Technology, Iran, in Fall 1993. She completed two masters degrees in industrial engineering (Summer 1996) and engineering science (Summer 2000) at Louisiana State University both under the supervision of Dr. Gerald M. Knapp. Currently she is working in the education services department of i2 Technologies Inc. based at their headquarters in Dallas, Texas.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Roya Javadpour

Major Field: Engineering Science

Title of Dissertation: A Neural Network Approach to Dependent Reliability Estimation

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

25 June 2001